Universidade do Sul de Santa Catarina

Metodologias e Projetos de Software

Disciplina na modalidade a distância

5ª edição revista e atualizada

Palhoça

UnisulVirtual

2008

Créditos

Unisul - Universidade do Sul de Santa Catarina
UnisulVirtual - Educação Superior a Distância

Apresentação

Parabéns, você está recebendo o livro didático da disciplina de **Metodologias e Projetos de *Software***.

Este material didático foi construído especialmente para este curso, levando em consideração o seu perfil e as necessidades da sua formação. Como os materiais estarão, a cada nova versão, recebendo melhorias, pedimos que você encaminhe suas sugestões sempre que achar oportuno via professor tutor ou monitor.

Recomendamos, antes de você começar os seus estudos, que verifique as datas-chave e elabore o seu plano de estudo pessoal, garantindo assim a boa produtividade no curso. Lembre: você não está só nos seus estudos. Conte com o Sistema Tutorial da UnisulVirtual sempre que precisar de ajuda ou alguma orientação.

Desejamos que você tenha êxito neste curso!

Equipe UnisulVirtual.

Sumário

Palavras da professora

Caro aluno a disciplina de Metodologias e Projetos de *Software* vai inserí-lo no universo da modelagem de projetos de *software*.

O caminho da modelagem através dos anos tem sido árduo, sofrendo inúmeras inclusões e alterações, tudo isto para se adaptar as constantes evoluções de novas linguagens de programação, bancos de dados, sistemas operacionais, regras de negócio e aos novos paradigmas da programação.

A comunidade de desenvolvimento percebe-se cada vez mais da necessidade de documentar seus projetos, o volume de linhas de código crescente torna nossas vidas dia a dia mais informatizadas. Mas, este benefício cobra seu preço, o *software* sofre manutenções, equipes são constantemente modificadas, mas mesmo assim é necessário manter este arsenal de códigos executando com eficácia e eficiência.

Este é um dos contextos mais relevantes da modelagem, permitir o entendimento do projeto a qualquer membro da equipe em qualquer ponto do processo, seja em uma etapa inicial de análise de requisitos ou já em fase de manutenção no cliente.

A modelagem pode ser comparada ao projeto de uma residência: se, por exemplo, o desejo fosse a construção de um pequeno canil e você estivesse seguro sobre como fazê-lo, é bem provável que seria possível executar esta obra sem a necessidade de se fazer um projeto, pois ele vai ser pequeno e não teria muitos riscos. Agora, imagine a construção de um edifício de 3 andares, será que você teria coragem de edificá-lo sem a ajuda de um bom projeto que indique todos os aspectos relevantes da obra?

Você neste caso, concorda comigo, acredita que não, não é mesmo? Pois então, nesta mesma relação, imagine a concepção de um projeto de *software*, tenha a convicção que a modelagem fiel do sistema proporcionará ganhos de qualidade e economia de recursos a longo prazo.

Então, convencido da importância do estudo desta disciplina? Você está pronto para iniciar os estudos?

Bons estudos!

Professora Vera Schuhmacher

Plano de estudo

O plano de estudos visa a orientá-lo no desenvolvimento da disciplina. Ele possui elementos que o ajudarão a conhecer o contexto da disciplina e a organizar o seu tempo de estudos.

O processo de ensino e aprendizagem na UnisulVirtual leva em conta instrumentos que se articulam e se complementam, portanto, a construção de competências se dá sobre a articulação de metodologias e por meio das diversas formas de ação / mediação.

São elementos desse processo:

- O livro didático.
- O Espaço UnisulVirtual de Aprendizagem (EVA).
- O Sistema tutorial.
- O Sistema de avaliação (complementares, a distância e presencial).

Ementa da disciplina

Análise de Requisitos. Introdução ao RUP (*Rational Unified Process*). O paradigma orientado a objetos. Análise arquitetural. Modelagem de um sistema utilizando-se a notação UML: modelagem de use cases, análise e design; realização de use-case, diagrama geral de classes persistentes, diagrama de interfaces e mapeamento objeto-relacional. Desenvolver o aluno de forma a capacitá-lo a utilizar metodologias orientadas a objeto para a modelagem de sistemas.

Créditos: 8

Objetivos

- Propiciar ao aluno o conhecimento sobre conceitos relacionados ao ciclo de vida de desenvolvimento de um *software*.
- Sensibilizar o aluno sobre a importância do uso de metodologias de projeto de *software* para o sucesso efetivo de seu projeto.
- Tornar presente a discussão sobre a relevância de dispensar tempo suficiente para as etapas iniciais do projeto como a análise de requisitos procurando uma maturidade na compreensão das necessidades do usuário.
- Oferecer ao aluno o arsenal didático necessário para compreender o uso de técnicas e métodos estruturados de modelagem de sistemas.
- Desenvolver o a aluno de forma a capacitá-lo a utilizar metodologias orientadas a objeto para a modelagem de sistemas.

Agenda de atividades / Cronograma

- Verifique com atenção o EVA, organize-se para acessar periodicamente a sala da disciplina. O sucesso nos seus estudos depende da priorização do tempo para a leitura, da realização de análises e sínteses do conteúdo e da interação com os seus colegas e tutor.
- Não perca os prazos das atividades. Registre no espaço a seguir as datas com base no cronograma da disciplina disponibilizado no EVA.
- Use o quadro para agendar e programar as atividades relativas ao desenvolvimento da disciplina.

Atividades obrigatórias	
Avaliação a distância (AD)	
Avaliação presencial (AP)	
Avaliação final (AF)	
Demais atividades (registro pessoal)	

Tenha por hábito, usar o quadro para agendar e programar as atividades relativas ao desenvolvimento da disciplina.

UNIDADE 1

1

Ciclos de vida de um processo de desenvolvimento de *software*

Objetivos de aprendizagem

- Compreender as características do produto de *software*.
- Perceber as diversas etapas do processo de desenvolvimento de um *software*.
- Definir estas etapas dentro de um modelo de desenvolvimento de projeto.

Seções de estudo

Para início de estudo

Desenvolver *software* é uma tarefa árdua e complexa. Lidar com esta complexidade significa compreender claramente os processos relacionados ao desenvolvimento do *software*.

Mas por que é importante? É preciso entender claramente o que e como deve ser cobrado em cada processo, o que pode ser executado em paralelo ou de forma seqüencial. Se você pensou em atividade de gerência, está correto! Conduzir o processo de desenvolvimento de *software* é um grande processo de gerência e para conduzi-lo da melhor forma possível, é preciso saber o que se espera de cada etapa.

Nesta unidade você vai perceber que todo o processo de desenvolvimento, por mais complexo que pareça, passa por etapas pré-definidas e universais. Essas etapas são conduzidas de forma diferente, dependendo da natureza do projeto ou mesmo da cultura de sua empresa. Definir a forma como esses processos serão executados é fundamental para o bom andamento de todo o processo e mesmo para a escolha de metodologias futuras.

Então? Quer iniciar esta jornada metodológica? Bom estudo!

SEÇÃO 1 - Quais são as características do *software*?

Termos como internet, *web* e *software*, são modernos e estão na mídia diariamente. São termos que antigamente eram exclusivos de áreas extremamente técnicas e que agora tornaram-se de domínio público.

Apesar de até mesmo parecer "feio" não saber seu significado durante uma conversa entre amigos, é importante verificar se você realmente conhece o verdadeiro significado. Você sabe realmente o que significa *software*?

Será que você sabe realmente o que significa *software*?

Explicar este conceito não é simples, depende da ótica pela qual se tenta entender.

Você poderia dizer que *software* é o conjunto de instruções que ao serem executadas produzem a função e o desempenho desejados.

Poderia conceituar *software* como estruturas de dados que possibilitam que os programas manipulem adequadamente a informação.

É possível dizer ainda que *softwares* são os documentos que descrevem a operação e o uso dos programas desenvolvidos ou projetados por engenharia, não manufaturados no sentido clássico.

Todavia é possível ser redundante e dizer que *softwares* são ferramentas pelas quais se explora os recursos do *hardware*, executa-se determinadas tarefas, se resolve problemas ao interagir com a máquina e tornar o computador operacional.

Conceituar *software* passa por conhecer suas características, as quais são relacionadas a seguir.

O *software* apresenta características únicas, ele não se desgasta mas se deteriora.

Mas o que é isto?

Imagine que você compre esta Ferrari. Novinha !!! Agora imagine você circulando com esta Ferrari todos os dias 200 Km em uma estrada de chão cheia de buracos.

Como vai estar sua Ferrari em um ano??

Com certeza vai estar cheia de ruídos e talvez apresente problemas em alguns componentes, isto acontece pelo desgaste do carro. Ao final de 10 anos usando o carro nesta mesma estrada é bem provável que você tenha um carro cheio de problemas!

E, se a gente for comprar um *software*, imagine que você tenha uma vídeolocadora, e resolva comprar um *software* para informatizar todo o processo de atendimento.

Abstraindo-se questões relacionadas ao *hardware*, este *software* daqui a 100 anos irá executar da mesma maneira que ele executa hoje. Ou seja, se tiver algum problema de programação irá repetir este problema em todas as vezes que for executado. Mas se o *software* não tiver problemas irá executar perfeitamente todas as vezes em que você solicitar sua execução.

Mesmo executando perfeitamente ao final de 3 anos talvez ele não supra mais suas necessidades, imagine o seguinte: no seu sistema você digita o código da fita para realizar a movimentação, mas agora gostaria que ele lesse o código de barras da fita. E agora ?

É por isto que dizemos que o *software* não se desgasta, mas se deteriora em função de inovações tecnológicas e novas necessidades do cliente. Ele acaba não atendendo mais suas necessidades apesar de funcionar, muitas vezes sem apresentar problemas.

O *software* é desenvolvido ou projetado por engenharia, não manufaturado no sentido clássico.

Quando você comprou sua geladeira, em algum momento pensou que ela foi fabricada como uma peça única?

Claro que não, ela foi produzida em uma linha de produção. Pense em uma linha de produção de caminhões. Os componentes são incorporados ao projeto para dezenas de unidades.

Mas quando se pensa em *software*, é sabido que cada produto é projetado e pensado como uma peça única. É impossível visualizá-lo em uma esteira de produção !

Agora use a imaginação, imagine as seguintes situações:

- Na linha de produção de caminhões, saem da fábrica 30 unidades diárias, 600 por mês. Destas 600 unidades a indústria possui uma estatística de problemas apresentados em 0,2% da produção durante sua utilização pelo cliente nos primeiros 6 meses. Isto significa uma margem de problemas em 1 caminhão saído da linha de produção mensal. Esta informação pode acabar com o nome da empresa ? O que você acha?

- Agora imagine uma empresa que desenvolve *software*, ela desenvolve nosso famoso sistema de vídeolocadora e vende este produto para 600 clientes. Mas no final do sexto mês acontece um erro no sistema (ele não está armazenando corretamente a data de entrega da fita). Sendo que esta versão foi entregue para os 600 clientes teremos então 600 clientes com o mesmo problema. E agora? Esta informação pode acabar com o nome da empresa ? O que você acha?

Sintetizando o problema: Em uma fábrica de *software* o sucesso é medido pela qualidade de uma única entidade e não pela qualidade de muitas entidades manufaturadas.

SEÇÃO 2 - Quais as etapas do processo de desenvolvimento de *software*?

Existem alguns conceitos que são fundamentais quando falamos de desenvolvimento, o primeiro deles é o termo Engenharia de *Software*. Assim como a palavra *software*, engenharia de *software* também é um termo que gera discussões sobre a melhor maneira de expor todas as suas nuances.

IEEE - Instituto de Engenharia Elétrica e Eletrônica, formado pela fusão do Instituto de Engenheiros de Rádio (IRA) com o Instituto Americano de Engenheiros Elétricos (AIEE). A meta do IEEE é promover conhecimento no campo da engenharia elétrica e eletrônica estabelecendo padrões para formatos de computadores e dispositivos.

A **IEEE** em 1993 definiu Engenharia de *Software* como a aplicação de uma abordagem sistemática, disciplinada e quantificável para o desenvolvimento, operação e manutenção

do *software*; o estudo de abordagens e princípios, a fim de se obterem economicamente *software*s confiáveis e que executem de forma eficiente nas máquinas reais.

Outro conceito bem aceito pela comunidade computacional é o de Krakowak (1985), exposto em um congresso em 1985. Para ele, Engenharia de *Software* (EGS) é o conjunto de métodos, técnicas e ferramentas necessárias à produção de *software* de qualidade para todas as etapas do ciclo de vida do produto.

Mas a partir desse conceito, é importante que você entenda o significado de método, procedimento e ferramenta, que são fontes de muita confusão, segundo PRESSMAN (2002):

- **Métodos** - proporcionam os detalhes de "como fazer" para construir o *software*. Os métodos envolvem um amplo conjunto de tarefas, como o planejamento e estimativa do projeto, a análise de requisitos de *software*, o projeto da estrutura de dados, o algoritmo de processamento, a codificação, o teste e a manutenção.
- **Ferramentas** - fornecem suporte automatizado aos métodos. Atualmente existem ferramentas para sustentar cada um dos métodos. Quando as ferramentas são integradas, é estabelecido um sistema de suporte ao desenvolvimento de *software*, chamado *CASE - Computer Aided Software Engineering*.
- **Procedimentos** - constituem o elo de ligação entre os métodos e ferramentas, eles estabelecem a seqüência em que os métodos serão aplicados, estabelecem os produtos (*deliverables*) que devem ser entregues, os controles que ajudam a assegurar a qualidade e coordenar as alterações e os marcos de referência que possibilitam administrar o progresso do *software*.

Quando se fala em produção de *software*, é muito difícil estabelecer o preço, prazo e número de pessoas necessárias para o desenvolvimento. Para isso, existe uma metodologia chamada **ponto por função**, que nos ensina, mediante seu método, a encontrar resultados para as dúvidas relacionadas a essas estimativas. Mas, como fazer isso manualmente é difícil, foram, então desenvolvidas ferramentas baseadas nesse método, que o automatizam (como o Gemétrics por exemplo, que foi desenvolvido por alunos de graduação). Mas **o ponto por função é uma metodologia que estabelece entrada de informações e resultados na forma de cálculos e relatórios**. Os procedimentos serão utilizados para dizer quando as entradas de informações serão feitas, quais relatórios serão emitidos e em que momento.

Segundo Pádua (2000), a EGS preocupa-se com o *software* como produto. E, como todo produto industrial, o *software* tem um ciclo de vida que:

- deve ser concebido a partir da percepção de uma necessidade;
- é desenvolvido transformando-se em um conjunto de itens entregue ao cliente;
- entra em operação sendo usado dentro de um processo de negócio e sujeito à manutenção quando necessário;
- é retirado de operação ao final de sua vida útil.

O que é processo de *software*?

Quando se fala em processo longo, nos vem a mente uma seqüência de passos para atingir um objetivo.

E, como afirma Pádua (2000), processo de *software* é exatamente isto: um conjunto de passos ordenados, constituídos por

atividades, métodos, prática e transformações, usado para atingir uma meta.

Como você já deve estar imaginando, processos podem ser definidos para atividades em todas as etapas do desenvolvimento de um *software*.

PRESSMAN (2002) sugere a divisão do desenvolvimento em três processos genéricos, acompanhe a figura:

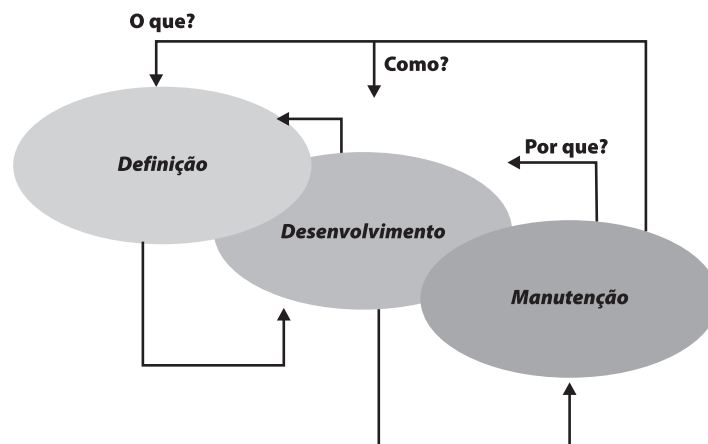



Figura 1.1 - Etapas do desenvolvimento de software
Fonte: PRESSMAN (2002).

Cada etapa pode ser detalhada como:

a) Definição

A primeira etapa, constitui-se por identificar quais informações devem ser processadas, qual função e desempenho são desejados, quais interfaces devem ser estabelecidas, quais restrições de projeto existem (ex: o cliente precisa do *software* em 60 dias !) e quais critérios de avaliação são exigidos para se definir um sistema bem sucedido (ex: existem normas internacionais que devem ser obedecidas no chão da fábrica e devem ser incorporadas ao sistema).

A etapa de definição se subdivide em:

- **Planejamento do projeto** – no planejamento avaliamos se o projeto é viável, e como o desenvolvimento será conduzido.

- **Levantamento de requisitos** – quando você está nesta etapa, o objetivo principal é compreender o problema do cliente, suas necessidades, expectativas e restrições. Aqui você deve pensar com o cliente: quais os problemas que existem? Como eles acontecem? Existe alguma característica específica que deve ser observada: de velocidade, portabilidade, interface?
- **Análise de requisitos** – nesta etapa, você vai analisar os dados da etapa anterior. Na análise de requisitos ou especificação de requisitos, vamos conceber uma estratégia para solucionar problemas e necessidades do cliente. Mas, nesta etapa, você ainda não precisa se preocupar com questões tecnológicas. Assim, pode-se afirmar que você vai dizer O QUE o sistema deve fazer para apoiar o cliente, a partir de modelos.

b) Desenvolvimento

Nesta fase, você tenta definir como a estrutura de dados e arquitetura do *software* têm de ser projetadas. Durante a segunda etapa, você irá realizar:

- **O projeto de** ***software*** – no projeto de *software*, você deve definir como o sistema vai funcionar para atender os requisitos levantados. Nesta fase, temos como resultado uma descrição computacional do que o *software* deve fazer; normalmente temos aqui o projeto da arquitetura (especificação da configuração de componentes do *software* - funções, classes, objetos e suas interconexões) e o projeto detalhado (tem por objetivo a concepção e especificação das estruturas de dados e dos algoritmos, que realizam aquilo que foi especificado para cada componente do *software*).
- **Implementação** – na implementação o sistema é codificado.
- **Teste de** ***software*** – Na etapa de testes ocorre a verificação do que foi implementado. Várias são as técnicas: desde corretivas (procura de erros) até de validação com grupos de usuários.

c) Manutenção

Nesta fase, ocorrerão mudanças no *software* em conseqüência dos erros encontrados. Além disto, esta é a etapa responsável pelas adaptações do *software* em função da evolução do hardware e necessidades do cliente.

A manutenção pode ser corretiva (o cliente encontrará defeitos no *software*), adaptativa (são modificações no *software* que acomodam mudanças relacionadas à evolução do hardware, por exemplo, o cliente decide trocar o sistema operacional por uma possibilidade open source, por exemplo) e funcional (implementa funções adicionais para o cliente, relacionadas a expectativas que ele não tinha durante o desenvolvimento, como em uma situação onde o gerente de recursos humanos precisa de um novo relatório estatístico).

Além das três etapas básicas, temos que considerar ainda as atividades complementares fundamentais para o bom andamento do processo:

- **Revisões** - efetuadas para garantir que a qualidade seja mantida à medida que cada etapa é concluída.
- **Documentação** - é desenvolvida e controlada para garantir que informações completas sobre o *software* estejam disponíveis para uso posterior.
- **Controle das Mudanças** - é instituído de forma que as mudanças possam ser aprovadas e acompanhadas pelos gerentes e pela equipe de projeto.

SEÇÃO 3 - Modelo de desenvolvimento de *software*

O processo de desenvolvimento apresentado na seção 2 é um modelo genérico, isto significa que suas etapas são aplicáveis a qualquer modelo, independente do paradigma de desenvolvimento utilizado. As diferentes formas de proceder em relação ao processo de desenvolvimento apresentam características próprias a cada modelo, procurando adaptar-se as diferentes necessidades das empresas desenvolvedoras de *software* ou mesmo ao tipo de sistema desenvolvido.

PRESSMAN (2002) define o modelo de desenvolvimento como uma representação abstrata do processo de desenvolvimento que define como as etapas relativas ao desenvolvimento de *software* serão conduzidas e inter-relacionadas para atingir o objetivo do desenvolvimento do *software*.

Existem alguns modelos que estão a muitos anos conosco, outros são muito recentes. Dê uma olhadinha, a seguir, nos mais conhecidos por sua utilização junto às empresas.

a) Modelo Cascata

No modelo cascata os sub-processos são executados em uma seqüência rígida. Assim cada sub-processo passa a ser um marco de controle. Este modelo exige uma abordagem sistemática, seqüencial ao desenvolvimento de *software*.

Isso acontece da seguinte forma, o desenvolvedor visita a empresa e inicia a etapa de levantamento de requisitos, após algumas sessões finaliza esta etapa de levantamento e inicia a etapa de análise.

Finalizada a análise inicia-se o processo de desenho do sistema, com todo seu projeto. Finalizada esta etapa inicia-se a implantação do código. Observe que não existe retorno a etapas anteriores, o modelo é inflexível as etapas são seqüências e não permitem regresso, no modelo em cascata o desenvolvedor ao finalizar o desenho do projeto só poderá modificá-lo na etapa de manutenção.

Esta visão burocrática dos sub-processos geram situações difíceis de serem resolvidas pelos analistas. Pense : projetos reais raramente seguem o fluxo seqüencial que o modelo propõe, no início de um projeto é difícil estabelecer explicitamente todos os requisitos, existe uma incerteza natural. E o maior problema de todos: a falta de visibilidade para o cliente! Somente na etapa de implantação o cliente terá contato com o *software*.

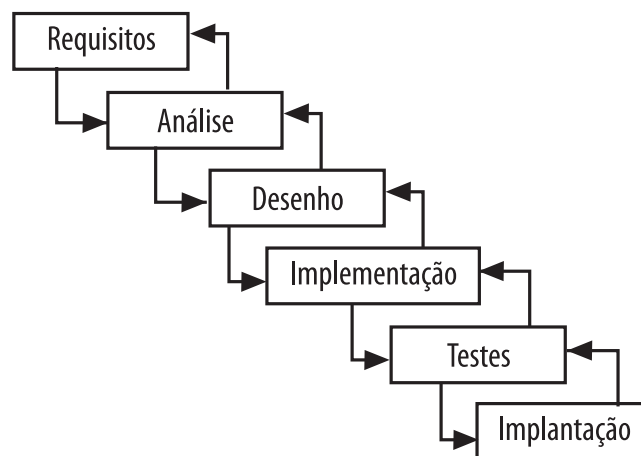



Figura 1.2 - Modelo Cascata
Fonte: Pádua (2000).

No modelo em cascata, todo o projeto deve ser entendido em suas fases iniciais, seguindo até o final com pouquíssima interação com o cliente.

Imagine a situação: você é contratado para desenvolver um projeto em uma fábrica de calçados, todo o processo de produção será automatizado. Em seu planejamento, você percebe que o processo todo estará finalizado em um prazo de 2 anos. Se você optar por este modelo somente no final de 2 anos, o cliente será apresentado ao sistema! Será que ele terá paciência para esperar por este prazo?

Outro ponto importante: o levantamento de requisitos é feito no início; e o processo segue para a etapa seguinte sem retornar à anterior; mas será que em 2 anos nada vai mudar no processo da empresa? Será, então, que este modelo é inviável para os tempos atuais? Saiba que este foi um dos modelos mais utilizados no mundo. Também é um modelo de baixo custo por sua estrutura seqüencial. Nos tempos atuais podemos sugeri-lo em situações onde você tem domínio do problema e/ou o projeto é considerado pequeno.

b) Modelo Espiral

O modelo em espiral é totalmente diferente do modelo anterior. Neste modelo a palavra de ordem é a experimentação e a avaliação.

O modelo é desenvolvido em uma série de interações; cada interação corresponde a uma volta na espiral. Cada volta é fundamentada na análise de riscos da solução que está sendo proposta.

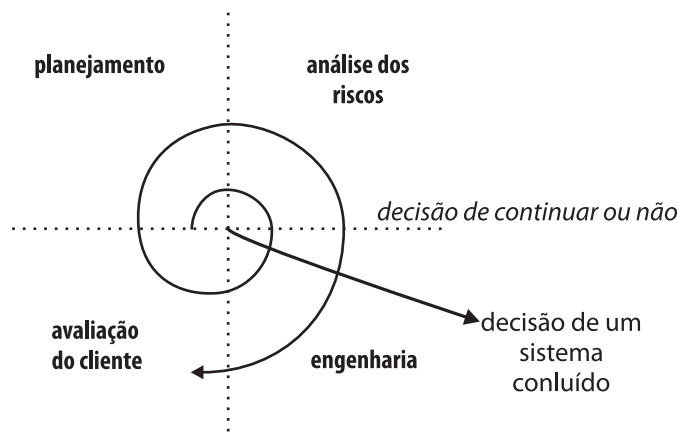



Figura 1.3 - Modelo Espiral
Fonte: Pressman (2002).

Na etapa de planejamento são determinados os objetivos, alternativas e restrições. Durante o sub-processo de análise de risco, são analisadas as alternativas e identificados os riscos e resoluções possíveis. Na construção ocorre o desenvolvimento do produto no nível seguinte. A avaliação do cliente é fundamental, pois nela ocorre a avaliação do produto e o planejamento das novas fases; cliente e desenvolvedor refinam os requisitos dos *softwares* a serem desenvolvidos.

O modelo exige dos desenvolvedores experiência na determinação de riscos, sendo, portanto, dependente da experiência pessoal da equipe.

Talvez você esteja se perguntando quando poderá utilizar este modelo. É uma boa pergunta. Imagine que você tenha que apresentar uma solução para um cliente e que você não esteja certo sobre a tecnologia a ser utilizada ou a forma como deva ser

construída a estrutura de dados. A melhor maneira de fazer o "melhor serviço" será usando o modelo espiral.

Utilize prototipação, simulação ou qualquer recurso possível para avaliar diferentes soluções até encontrar a melhor solução! Quando você estiver certo da resposta siga em direção da engenharia, onde temos as etapas padrão do processo de desenvolvimento.

Imagine que você foi contratado para desenvolver um produto que faça vídeo inspeção a distância de dutos de galerias de drenagem para a prefeitura do Rio de Janeiro. Você poderia de imediato indicar uma solução? Ou você teria dúvidas sobre como fazê-lo da melhor maneira, pela grande oferta de recursos tecnológicos e mesmo restrições de projeto existentes?

____

____

____

____

____

c) Modelo Prototipação

A prototipação envolve a produção de versões iniciais - "protótipos" - de um sistema futuro com o qual podem-se realizar verificações e experimentações para se avaliarem algumas de suas qualidades antes que o sistema venha realmente a ser construído. Este modelo é popularmente usado como um mecanismo para identificar os requisitos de *software*.

Imagine você iniciando um trabalho com um cliente que lhe solicita um *software* para controle de uma UTI; mas você não faz idéia do funcionamento de uma UTI. E o pior é que o cliente não parece dispor de muito tempo para explicar novamente o que você não conseguiu entender.

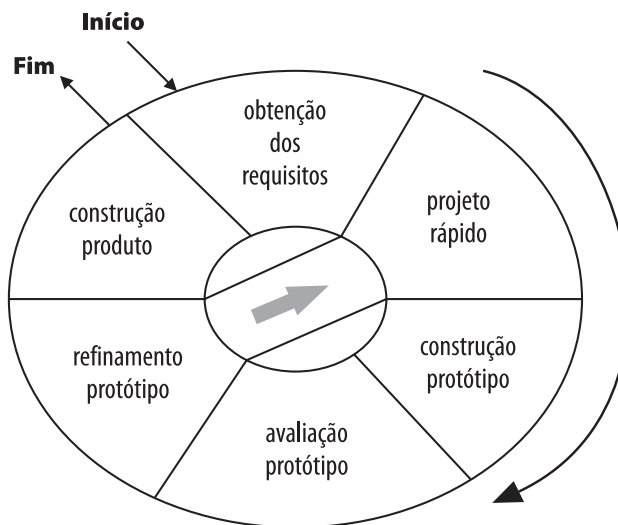



Figura 1.4 - Modelo Prototipação
Fonte: Pressman (2002)

Na figura 1.4 você percebe o ciclo de desenvolvimento da prototipação, o processo inicia com a obtenção de requisitos, segue-se então um projeto rápido para o futuro desenvolvimento do protótipo. Finalizado o protótipo é realizada uma avaliação com o cliente ou com a própria equipe de projeto. Os resultados da avaliação são utilizados para refinar o protótipo reiniciando o ciclo até a avaliação. Este processo se repete até o momento em que o grau de aceitação do protótipo seja considerado aceitável. A partir deste momento inicia-se a construção do produto. Na construção do produto pode-se utilizar qualquer outro modelo, cascata, incremental; entre outros.

A prototipação pode ser conduzida segundo duas técnicas específicas de construção. A prototipação horizontal, em que uma camada específica do sistema é construída (a interface do usuário com suas janelas e *widgets* ou camadas da aplicação, como as funções para transação em bancos de dados).

Ou a prototipação vertical, em que uma parte da funcionalidade do sistema é escolhida para ser implementada completamente. A prototipação vertical é interessante quando aspectos da funcionalidade não estão claros. A partir da validação do protótipo o modelo segue no processo tradicional de desenvolvimento para a construção do produto.

Lembre-se: quando se valida uma informação verbal com o cliente, em muitos casos a atenção do cliente não é total, e muitas palavras se perdem durante o diálogo. Um desenho no papel do protótipo de uma tela torna-se 100 vezes mais claro e objetivo, além de conseguirmos 100% da atenção do cliente.

d) Modelo Incremental

O Modelo Incremental foi desenvolvido através da combinação entre os modelos linear e prototipação (Pressman, 2002). Quando você usa este modelo todo o desenvolvimento é dividido em etapas que são produzidas de forma incremental até se chegar a um sistema finalizado.

Para cada etapa é realizado um ciclo completo de desenvolvimento e novas funcionalidades são adicionadas ao sistema. Assim, você pode dizer que todo o desenvolvimento evolui em versões.

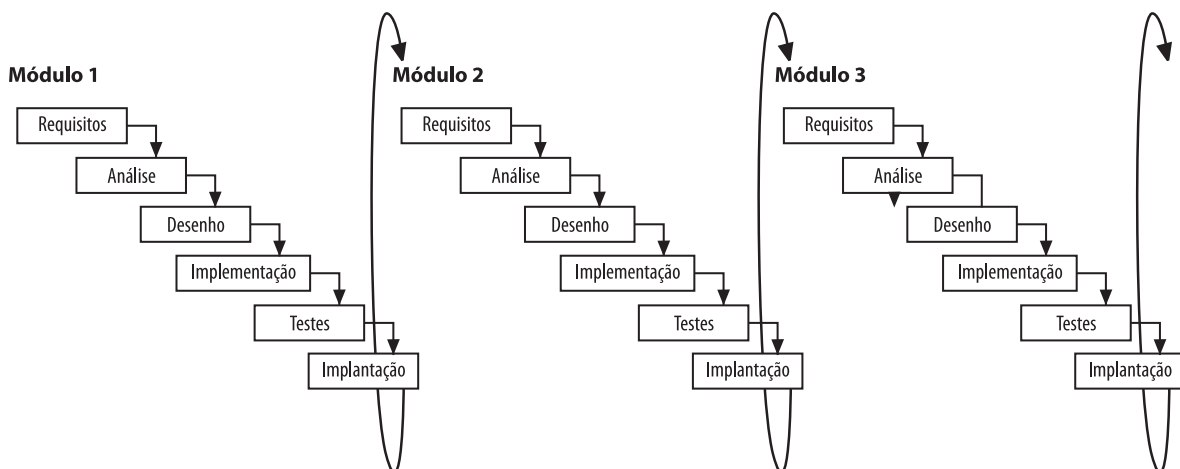



Figura 1.5 - Modelo Incremental
Fonte: PÁDUA (2000).

Se você deseja utilizar este modelo, considere a forma como todo o projeto será construído. Pense em questões relacionadas à prioridade (o que é mais importante para o cliente) e o risco de cada requisito. Tenha estratégia em mente e observe se o que é prioritário pode ser desenvolvido primeiro. Não esqueça, o primeiro módulo é o cartão de visitas de sua empresa para o cliente!

Neste modelo temos uma participação forte do cliente no projeto que avalia os incrementos entregues, o que propicia que você possa corrigir problemas nos módulos em desenvolvimento e realize os acertos devidos. Outro ponto importante é que você deve considerar, logo no início, os requisitos mais arriscados. Fazendo isto, qualquer inconsistência aparece logo no início, permite que você tenha tempo de reagir solucionando o problema.

Fundamental neste modelo é a visão global do sistema; o gerente deve ter em mente a soma de todos os módulos, evitando redundâncias na construção do produto.

O modelo incremental é um dos mais utilizados na atualidade; em sua estrutura é comum percebermos a inserção da prototipação.

E as Metodologias Ágeis?

Até pouco tempo, tudo o que se pensava sobre metodologias de desenvolvimento de software parecia extremamente ligado a farta documentação e processos claramente delineados. Para muitos desenvolvedores no entanto passos firmemente delineados e documentados para cada etapa do desenvolvimento eram considerados como uma burocracia desnecessária e que por vezes produzia altos custos ao projeto. A divergência sobre a eficiência do uso de modelos que consideravam modelos tradicionais como limitadores para equipes de desenvolvimento e não raro eram apontados como causa de atrasos e dificuldades no desenvolvimento. Por outro lado, pequenas empresas a margem de grandes orçamentos, ficavam a margem dos modelos tradicionais por sua incapacidade de comportar a estrutura necessária para sua execução.

A demanda criada a partir desta situação deu origem a um novo modelo, segundo Soares (2004), voltado a pessoas e não a processos ou algoritmos: os métodos ágeis. As metodologias ágeis possuem características que determinam um foco maior na codificação e não na documentação, são claramente adaptativas pois novos fatos que são apresentados no decorrer do projeto são

tratados durante o desenvolvimento, não existe a preocupação de conhecer o todo a partir de uma avaliação preditiva: novidades são adaptadas e incorporados durante o desenvolvimento. A metodologia ágil é, portanto, naturalmente iterativa incremental e acrescenta uma boa participação do usuário no processo.

A seguir conheça duas representantes desta nova corrente: as metodologias *Extreme Programming* e a SCRUM.

***Extreme Programming* (XP)**

O método XP foi criado por Kent Beck na década de 90, no ano de 1996 tornou-se mundialmente conhecido ao ser utilizado na empresa Daimler Chrysler em um projeto que durante anos não fora concluído e que ao fazer uso da metodologia teve sua execução completa realizada em apenas 4 meses.

Mas o que é o XP? Segundo Beck (1999), o XP é uma metodologia ágil para equipes pequenas e médias que desenvolvem software baseado em requisitos vagos e que se modificam rapidamente.

Neste processo, a burocracia deve ser mínima, as equipes pequenas (sugere-se até 10 pessoas) que trabalham de forma incremental são apoiados integralmente pelo futuro usuário do sistema. A análise dos requisitos ocorre a medida que os mesmos são descritos e solicitados pelo usuário, isto quer dizer que o conhecimento sobre o problema evolui durante o desenvolvimento.

O XP é norteado por valores, princípios e regras. Beck (1999) define 4 valores que descrevem o XP, a comunicação, a simplicidade, o feedback e uma boa dose de coragem. A seguir acompanhe uma breve descrição destes valores segundo WUESTEFELD (2001).

- **Simplicidade** – está ligada a simplicidade do software, do processo usado no desenvolvimento, na comunicação e em tudo o que norteia o trabalho de desenvolvimento, ou seja, descomplicar , a regra do XP é simplificar !
- **Comunicação** – ela deve ocorrer em todos os sentidos, desde o uso dos escritos a partir da convenção de padrões de codificação para facilitar o entendimento até

a preferência por reuniões (presenciais) para discussão de requisitos e dúvidas, o trabalho em salas comuns evitando o isolamento da equipe, a execução de revisões do projeto em conjunto com toda a equipe.

- ***Feedback*** – os possíveis problemas devem ser identificados o mais cedo possível para que possam ser corrigidos rapidamente. Da mesma forma oportunidades descobertas devem ser aproveitadas rapidamente.
- **Coragem** - ela é fundamental para apontar problemas, solicitar ajuda, para alterar e simplificar um que já está funcionando, para apresentar prazos que muitas vezes podem não agradar o seu usuário final, mas que são necessários para não comprometer a qualidade do projeto.

Os valores do XP são fundamentados em alguns princípios como o feedback rápido, a simplicidade que deve ser assumida em todas as etapas do projeto, a consciência de mudanças que devem acontecer de forma incremental, a compreensão e aceitação das mudanças e a necessidade da qualidade do trabalho.

Valores e princípios se entrelaçam na definição de práticas básicas que são adotadas pelo XP (Bona, 2002).

As doze práticas básicas que são adotadas pelo XP segundo SOARES (2004) são:

- **Jogo de Planejamento**: nesta prática decide-se o que deve ser feito (requisitos) e as prioridades. Nesta prática também é determinado o escopo da próxima versão. Dividido em duas áreas, o jogo do planejamento que decide sobre o escopo, a composição das versões e as datas de entrega é realizado pela área de negócios. Já as estimativas de prazo, o processo de desenvolvimento e o cronograma são determinados pelos programadores.
- **Programação em pares**: é produzido em duplas, dois programadores trabalham sempre juntos na mesma máquina. Enquanto um dos programadores escreve, o segundo confere a padronização e o raciocínio lógico.

- **Semana de 40 horas**: no modelo a semana é de 40 (quarenta) horas. Hora extra é abolida, parte-se do princípio que um programador cansado produz mais erros, e portanto, deve ser evitado.
- **Pequenas versões**: o sistema é baseado em entregas freqüentes a partir de versões com tamanho pequeno que são incrementadas em requisitos continuamente (entregas de versões mensais com feedback de aceitação do cliente, por exemplo).
- **Metáforas**: o uso de metáforas procura descrever o software sem a utilização de termos técnicos facilitando a comunicação com o cliente.
- **Projeto simples**: o sistema precisa ser projetado o mais simples possível satisfazendo os requisitos atuais, não havendo preocupação com requisitos futuros.
- **Teste**: a equipe de desenvolvimento descreve em um primeiro momento os casos de testes, após esta tarefa continuam o desenvolvimento. Assim como os desenvolvedores, os clientes também escrevem testes a fim de validar o atendimento dos requisitos.
- ***Refactoring***: sempre que possível o projeto deve ser aperfeiçoado, este aperfeiçoamento deve propor a clareza do software
- **Propriedade coletiva**: pertence a todo o grupo. Isto significa que não existe o conceito de propriedade para um algoritmo, qualquer integrante da equipe pode acrescentar ou alterar, desde que realize os testes necessários para validá-lo.
- **Integração contínua**: a construção e a integração do sistema ocorre várias vezes por dia. Tarefas são completadas continuamente.
- **Cliente dedicado**: o usuário final esta disponível todo o tempo, determinando os requisitos, atribuindo prioridades, e principalmente respondendo as dúvidas.
- **Padronização do código**: o código é escrito com um grande cuidado quanto a padronizações, isto facilita o entendimento assegurando a clareza e a comunicação entre programadores e projetistas .

SCRUM (*Software Development Process*)

Segundo relatos de Sutherland (2004), o SCRUM foi documentado e concebido na empresa *Easel Corporation* em 1993. A idéia inicial era voltada para empresas automotivas, mais tarde a metodologia foi incorporada pelas empresas *Advanced Development Methods* e VMARK passando por modificações e melhorias até chegar em um processo considerado adequado para projetos e desenvolvimento de *software* orientado a objetos.

O SCRUM possui aspectos que se assemelham ao XP, assim como o XP é formatado para o uso de equipes pequenas onde os requisitos não são claros e com interações de curta duração com um máximo de visibilidade no processo de desenvolvimento.

O SCRUM divide o desenvolvimento em interações com uma duração de 30 dias que são chamadas de *sprints*. As equipes multidisciplinares são formadas por projetistas, programadores, engenheiros e gerentes de qualidade. Cada equipe trabalha em torno dos requisitos do projeto que são definidos no início de um *sprint*. A cada dia ocorre uma reunião da equipe, nela discutem-se metas, dificuldades e objetivos, observe que isto acontece todos os dias.

Quais são as características da metodologia SCRUM?

Schwaber (1997) define algumas características da metodologia SCRUM que caracterizam etapas do seu processo de desenvolvimento.

Existem duas fases bem definidas: a primeira - planejamento (pré-game phase) e a segunda, fechamento (post-game phase) onde os processos são definidos claramente.

Durante o planejamento definem-se requisitos, são estabelecidas prioridades e estimativas, são feitas análises de necessidades como treinamento da equipe e ferramentas. Na fase de *sprint* (game phase) todo o processo baseia-se na experiência da equipe. As fases do *sprint* são gerenciadas por meio de controles que abrangem riscos e a otimização de questões de flexibilidade.

Os *sprints* não apresentam uma estrutura linear, muitas vezes o uso da tentativa e do erro é utilizado para determinar e reconhecer ou conhecer o processo.Ainda assim, é importante entender que o sprint acontece de forma tradicional, ou seja, seguindo etapas de análise, projeto, implementação e testes. A característica forte do SCRUM é o processo rápido, os ciclos duram de 7 a 30 dias.

Todo o desenvolvimento cresce de forma incremental podendo ser alterado durante todo o processo de desenvolvimento.

O fechamento do projeto baseia-se no ambiente e pode-se dizer que o produto que será entregue será determinado durante o projeto. A finalização (post-game phase) apresenta o produto ao cliente e procura avaliar com a equipe, a evolução do projeto. Nesta etapa questões como testes, integrações e documentação são tratadas.

Quer conhecer mais sobre esses modelos?

Acesse os *links*:

Modelo XP *Programming* – apresenta uma abordagem de programação em pares, com a definição da etapa de testes no início do projeto e intensa participação do usuário final
http://www.extremeprogramming.org/

Qual o melhor modelo, qual é o pior? Estas definições dependem da natureza do projeto, da empresa e sua organização, das ferramentas e recursos existentes na empresa de desenvolvimento.

Uma idéia que ganha cada vez mais adeptos é a combinação de diferentes modelos aproveitando o melhor de cada um.

Agora, para praticar os conhecimentos conquistados nesta unidade, realize as atividades propostas.

Atividades de auto-avaliação

Leia com atenção os enunciados e, após, realize as questões propostas:

1) Classifique as questões a seguir, em Verdadeira (V) ou Falsa (F).

a) (   ) O *software* pode ser definido como um conjunto de instruções se visto sob o ponto de vista do programador.

b) (   ) Uma das características mais interessantes do *software* está relacionada ao fato de ser um produto personalizável, isto se deve por ser um produto de engenharia e não manufaturado no sentido clássico.

c) (   ) O *software* e a empresa que o produz são muitas vezes avaliados por apenas uma unidade, pois a mesma pode ser vendida para dezenas de clientes com ou sem customização.

d) (   ) O desgaste do *software* refere-se apenas à adaptação de produtos a novas tecnologias.

e) (   ) Uma definição completa de *software* pode ser sugerida a partir da soma de instruções, documentação e estrutura de dados.

2) Relacione a primeira coluna com a segunda.

A. Definição

B. Levantamento de requisitos

C. Desenvolvimento

D. Projeto de *software*

E. Manutenção

F. Revisões

G. Atividades de apoio

H. Análise de requisitos

( ) Composto por atividades que ocorrem no decorrer de todo o processo de desenvolvimento, como revisões, controle de mudanças e documentação.

( ) Define o que deve ser feito para apoiar o cliente em seus problemas.

( ) Nesta etapa, são executados o projeto, codificação e testes do *software*.

( ) Etapa que identifica as necessidades do cliente.

( ) É a etapa responsável por alterações de cunho corretivo e adaptativo do *software*.

( ) Atividades efetuadas para garantir que a qualidade seja mantida.

( ) Fazem parte desta etapa o planejamento do projeto, levantamento e análise de requisitos.

( ) Define como o *software* irá implementar a solução do cliente.

3) Assinale as afirmativas corretas:

a) ( ) Os métodos proporcionam detalhes relacionados à construção do *software* para as etapas de planejamento e estimativa.

b) ( ) Os métodos estabelecem detalhes de como fazer para construir um *software* em todas as etapas do processo; as ferramentas automatizam os métodos, facilitando sua utilização; os procedimentos estabelecem controles, artefatos que assegurem a correta utilização do método.

c) ( ) Procedimentos proporcionam os detalhes de como construir o *software*; ferramentas automatizam os procedimentos; os métodos formam o elo de ligação entre ambos.

4) Relacione as características de cada modelo [**C**]ascata, [**P**]rototipação, [**I**]ncremental ou [**E**]spiral:

a) (   ) Neste modelo de desenvolvimento é possível avaliar riscos de projeto, tomando-se decisões baseadas na experimentação de diferentes soluções.

b) (   ) O modelo facilita a identificação de requisitos para a construção do futuro *software* por meio de protótipos.

c) (   ) O sistema é fragmentado, sendo que cada fragmento percorre todo o ciclo de desenvolvimento do *software*.

d) (   ) Neste modelo, as sub-tarefas são executadas de forma seqüencial e bastante rígida, sendo que o cliente tem contato com o sistema somente quando ocorre sua conclusão.

5) Assinale com "x" qual dos modelos a seguir oferece menor contato com o cliente?

a) (   ) Modelo cascata

b) (   ) Modelo incremental

c) (   ) Modelo prototipação

d) (   ) Modelo espiral

6) A empresa CONSTONÓIS procura sua empresa para solicitar o desenvolvimento de um *software*, para controle de vendas de imóveis. A empresa responsável pela construção de prédios realiza a venda de seus imóveis na forma a vista ou por financiamento direto com a construtora. O sistema deve possibilitar o controle das vendas assim como o controle do financiamento, que aceita ouro, carros, imóveis consórcios, dólares ou reais como parte do pagamento. O cliente não deixou muito claro o cálculo do saldo devedor do comprador nem a forma como ocorre o acúmulo de débito do cliente. O sistema apresenta, em sua análise inicial, 30 funcionalidades entre cadastros e relatórios de consulta. Qual o modelo de desenvolvimento que você adotaria?

a) (   ) Modelo cascata

b) (   ) Modelo incremental

c) (   ) Modelo prototipação

d) (   ) Modelo espiral

7) Observe o modelo XP e o modelo SCRUM, e a seguir descreva o que é possível determinar como diferenças fundamentais em relação aos modelos tradicionais.

Síntese

Nesta primeira unidade você teve contato com conceitos e modelos relacionados ao processo de desenvolvimento de *software*. Também estudou sobre a importância de se estabelecer claramente, nas empresas de *software*, os sub-processos existentes no processo de desenvolvimento.

Foi possível perceber que diferentes modelos servem para o desenvolvimento de sistemas com características específicas. O uso dos modelos também pode ser feito de forma combinada para um mesmo projeto. O projeto pode respeitar o modelo incremental, fragmentando suas funcionalidades; para alguns módulos em que não estão claros requisitos, utiliza-se então a prototipação; para um módulo onde existe a necessidade de avaliar algum risco; utiliza-se o modelo espiral. Misturando-se os modelos tem-se o melhor de todos eles.

Na próxima unidade você vai estudar a etapa de definição do processo de desenvolvimento de *software*. Esta etapa é o marco fundamental de seu projeto, pois é nela que você percebe o escopo, as necessidades e restrições que serão trabalhadas, conciliadas e solucionadas durante todo o ciclo de desenvolvimento do *software*.

Saiba mais

Para aprofundar as questões abordadas nesta unidade, você poderá pesquisar os seguintes livros:

- PRESSMAN, Roger. **Engenharia de** *Software*. São Paulo: McGraw-Hill, 2002.
- SOMMERVILLE, Ian. **Engenharia de** *software*. 6 ed. São Paulo: Addison-Wesley, 2003.

UNIDADE 2

2

Engenharia de requisitos

Objetivos de aprendizagem

- Reconhecer a importância da análise de requisitos no processo de desenvolvimento.
- Perceber as diferentes etapas que fazem parte da análise de requisitos.
- Realizar de forma consistente a atividade de análise de requisitos.

Seções de estudo

Para início de estudo

Quando você inicia um projeto de *software*, existe sempre uma preocupação: fazer o melhor projeto. Mas qual é o melhor projeto? Na verdade, é aquele que atende o cliente da melhor maneira.

A etapa de análise de requisitos é o momento onde as necessidades são exploradas por parte do desenvolvedor em companhia do cliente e sua empresa. Para que esta etapa finalize com uma proposta de solução adequada é necessário que tarefas, como o estudo de viabilidade, elicitação e análise de requisitos a especificação e o gerenciamento de requisitos sejam realizadas de forma cuidadosa e consistente.

Nesta unidade você vai perceber que a análise de requisitos é um processo interativo que envolve a compreensão do domínio, a coleta de requisitos, a estruturação e o estabelecimento de prioridades para a execução do projeto.

Você acredita que os requisitos são persistentes ao tempo? Será que um projeto iniciado em janeiro terá os mesmos requisitos ao final de 12 meses? A compreensão do projetista sobre o problema dispensa a participação do cliente em etapas de verificação? Com esta unidade você vai encontrar apoio para responder a algumas destas questões.

SEÇÃO 1 - O que é análise de requisitos?

Você já esteve envolvido na construção de uma casa? Ou teve alguém próximo a você construindo? Imagine então que você vai construir uma casa e deseja que seja espaçosa, com três quartos, três banheiros, de dois pavimentos, etc. Qual seria a primeira ação a ser tomada? Pense um pouco a respeito.

Bom, a primeira coisa que você vai com certeza fazer é contratar o pedreiro, comprar o material e iniciar a obra, pois você sabe exatamente como deve ser feita. Não é verdade? Não? Por quê?

Isto não acontece porque se sabe da necessidade de ter um projeto claro, em projetar os aspectos relacionados à estrutura hidráulica, elétrica e ter profissionais aptos a conceber a melhor solução para as nossas necessidades.

Finalizados os projetos, inicia-se então a obra e, neste momento, o projeto documentado das plantas funciona como roteiro de gerenciamento, ou seja, tudo o que for construído deve ser consistente com o que foi projetado.

Isto significa que se valida o que os pedreiros e mestres-de-obra realizam com o projeto feito por engenheiros e arquitetos. E, se você construísse sem fazer tais projetos, você acredita que sua casa seria construída dentro de suas expectativas, solucionando suas necessidades? É quase certo que isto não ocorreria!

Este exemplo também é valido para a construção do *software*, mas infelizmente muitas empresas ainda apostam no desenvolvimento do produto sem passar formalmente pela etapa inicial de definição e projeto, indo direto para a programação. Porém, o resultado é muitas vezes desastroso.

Ao se iniciar um projeto de *software*, inicia-se uma série de processos, marcados por entregas de artefatos pré-determinados pelos modelos de desenvolvimento utilizados e suas metodologias.

A etapa de análise de requisitos se encontra estrategicamente no início do projeto, pois é nessa etapa que você vai compreender as necessidades de seu futuro cliente.

Peters (2000) declara com grande propriedade que o grau de compreensibilidade, precisão e rigor da descrição, fornecida por um documento de requisitos de *software*, tende a ser diretamente proporcional ao grau de qualidade do produto resultante.

Mas o que é um requisito?

Requisito de *software* é uma descrição dos principais recursos de um produto de *software*, seu fluxo de informações, comportamento e atributos.

PÁDUA (2000) descreve que requisitos são características que definem os critérios de aceitação de um produto. Estas características podem ser divididas em:

- **características funcionais** – que representam o comportamento que o sistema deve apresentar diante das interações do usuário (o cliente deseja que seu vendedor lance o pedido de vendas para emitir relatório de comissões mensais por vendedor).
- **características não-funcionais** – representam aspectos comportamentais do sistema (o cliente deseja que o pedido seja lançado no momento que ele é feito pelo cliente, o que pode indicar a necessidade de recursos *wireless*).

Requisitos não-funcionais na maioria das vezes não são solicitados pelo cliente, mas devem ser inerentes ao projeto, são questões como a portabilidade do sistema, sua segurança e confiabilidade dos dados.

Você ainda pode explicar requisitos a partir do conceito de que o requisito é formado por :

- **requisitos explícitos** - são as necessidades ou as próprias condições e objetivos propostos pelo cliente (o cliente deseja ter os dados cadastrais de seus fornecedores);
- **requisitos implícitos** - incluem as diferenças entre os usuários, a evolução no tempo, as implicações éticas, as questões de segurança e outras visões subjetivas (o cliente deseja um site de comércio eletrônico; questões de segurança talvez não sejam sua preocupação, por sua falta de conhecimento em tecnologia, mas são requisitos que devem estar implícitos no seu produto);
- **requisitos normativos** - são decorrentes de normas, leis ou padrões (a emissão de uma nota fiscal deve seguir as regras propostas pela federação).

Na etapa inicial da análise de requisitos, é fundamental que o analista entenda as necessidades do cliente. Isso significa compreender claramente os requisitos explícitos, implícitos e normativos.

Essa compreensão da amplitude do problema é proporcional ao sucesso do projeto junto ao seu cliente final. Parece fácil, mas uma das maiores dificuldades neste processo é a dificuldade da comunicação entre Cliente x Desenvolvedor.

Veja a figura a seguir, a visão do que o cliente solicitou ao desenvolvedor, sua necessidade fica totalmente comprometida por falhas no entendimento e na comunicação. O resultado do "que acaba sendo feito" tem gerado custos elevados para empresas desenvolvedoras e para seus clientes que em muitas situações não conseguem usar o produto desenvolvido.

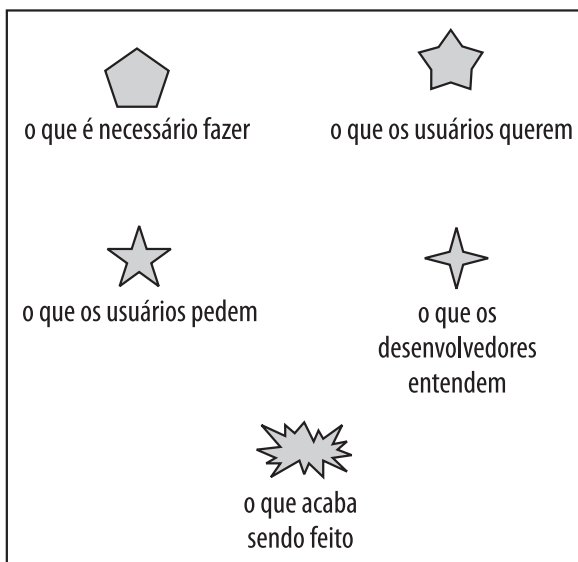



Figura 2.1 - Evolução dos requisitos
Fonte: Pádua (2002).

A dificuldade de entender o cliente é um desafio que deve ser encarado a partir de técnicas, métodos e paciência por parte do desenvolvedor.

Lembre-se: a pressa para chegar a implementação não será o caminho mais curto.

Os processos usados durante a engenharia de requisitos variam, dependendo do domínio da aplicação, das pessoas envolvidas e da organização que está desenvolvendo os requisitos.

Existem algumas atividades genéricas comuns a todos os processos, que são:

- Levantamento de requisitos
- Documentação de requisitos

- Especificação de requisitos
- Validação de requisitos
- Gerenciamento de requisitos

Nas próximas seções, você vai estudar sobre as primeiras etapas do processo de engenharia de requisitos, começando pela etapa de levantamento de requisitos.

SEÇÃO 2 - Levantamento de requisitos

A etapa de levantamento de requisitos é a etapa onde ocorre a compreensão do problema aplicada ao desenvolvimento de *software*.

Quando você está nesta etapa é fundamental que usuários e desenvolvedores tenham a mesma visão do problema a ser resolvido. Isto significa que você precisa conhecer os requisitos tão bem quanto o próprio cliente. Mas como obter estes requisitos?

Durante o levantamento de requisitos você vai se deparar com um grande volume de relatórios, formulários e documentos. Mas, quais são os que você deve avaliar?

Por outro lado, em uma empresa com 500 funcionários, serão 500 as pessoas envolvidas no processo de levantamento, você vai entrevistar todas?

É nessa hora que você deve perceber a importância de "ver a floresta em vez das árvores". Detecte as pessoas-chave do processo, trabalhe usando amostragens da população. Escute com atenção a gerência da empresa e seus objetivos.

Retome o exemplo da construção de uma casa. Para que o arquiteto inicie o projeto ele precisa perceber o perfil do cliente, suas preferências e necessidades. E para esta etapa, existem algumas técnicas úteis: a entrevista, observações e a investigação.

a) Entrevista

O uso da entrevista é feito pelo uso do formato "pergunta-resposta". Usando esta técnica você pode obter opiniões do usuário, descobrir o que o cliente pensa sobre o sistema atual, obter metas organizacionais/pessoais e levantar procedimentos informais.

Quando você realizar uma entrevista, lembre-se de:

- tentar estabelecer com o cliente um clima de confiança e entendimento;
- manter-se sempre no controle da entrevista;
- tentar mostrar ao cliente sua importância dentro do sistema;
- se preparar antecipadamente para a entrevista.

Mas o que significa preparar-se para a entrevista?

Estude o material previamente, verifique o linguajar utilizado (imagine que, se o *software* for da área jurídica, você precisa falar e entender as nuances e significados dos termos utilizados durante sua entrevista), perceba aspectos relacionados à organização e mesmo sobre o futuro entrevistado. Estabeleça claramente os objetivos, saiba o que perguntar, não faça rodeios.

Lembre-se: inclua em sua lista de entrevistados pessoas-chave dentro do futuro sistema.

Quando você realizar uma entrevista, marque a data e a hora com antecedência, com duração de no mínimo 45 minutos e, no máximo, de duas horas. Elabore as questões e a estrutura da entrevista, e durante a entrevista registre tudo o que for possível, fazendo uso de anotações ou de um gravador.

Ao formular as questões tenha algumas precauções, como:

- evite usar questões que levem o entrevistado a responder de uma forma específica ou tendenciosa.

Inadequada: "Você também acredita que a prioridade do desenvolvimento deva ser o faturamento como seu gerente afirmou?"

Melhor: "O que você acha que deva ser implantado em primeiro plano?"

- fazer duas questões em uma, torna a pergunta confusa e a resposta pode não ser completa. Ainda é possível que a pessoa acabe respondendo a uma das questões apenas.

Inadequada: "Em que situações você cancela uma nota fiscal? Quando ocorre o cancelamento de uma nota fiscal, quais os procedimentos?"

b) Questionário

O questionário é uma técnica que permite o levantamento de informações a partir da coleta de informações de diferentes pessoas afetadas pelo sistema.

Segundo SOMMERVILLE (2000), nesta técnica são abordadas questões relacionadas ao que as pessoas na organização querem, ao que as pessoas consideram como verdade, ao comportamento das pessoas, às características das pessoas, procedimentos e equipamentos são levantados.

Um questionário deve ter questões claras e não ambíguas, ter fluxo bem definido, ter administração planejada em detalhes e ainda levantar antecipadamente as dúvidas das pessoas que irão respondê-lo.

Uma dica interessante é aplicar o questionário em uma amostra de usuários, solicitando a avaliação sobre a estrutura e o conteúdo do mesmo.

Sempre que possível use o vocabulário das pessoas que irão responder. Prefira o uso de perguntas curtas e simples. Certifique-se de que as questões estão tecnicamente precisas antes de incluí-las no questionário.

c) Observação Direta

A observação direta pode ser utilizada como validação das entrevistas, identificação de documentos, esclarecimento sobre o que está sendo realizado no ambiente atual e a forma como ocorre.

No mecanismo de observação direta o analista observa sem intervir diretamente no processo. É importante planejar a observação e isto significa identificar o que deve ser observado, obter aprovação das gerências apropriadas, obter as funções e nomes das pessoas envolvidas nas ações que serão observadas. Se você optar por esta técnica prepare os usuários com cuidado, esclarecendo a forma como o processo vai ocorrer.

d) Brainstorming

No sentido exato da palavra, brainstorming é uma tempestade de idéias. O uso da discussão em grupos em que, a partir dos resultados das técnicas acima, procura-se compreender corretamente documentos, respostas oferecidas pelos usuários, processos existentes, são a base para que se chegue a uma boa especificação.

Nesta etapa inicia-se a formatação de um documento que deve conter os requisitos necessários ao projeto dentro de um consenso entre desenvolvedores e cliente.

Durante o levantamento dos requisitos tembém é estabelecido o escopo do projeto e também as possíveis restrições que possam delinear algum tipo de risco no horizonte.

Imagine que, durante a entrevista, o cliente diga para você que não pretende investir em equipamentos e em novos produtos, como um banco de dados. Talvez esta informação se torne uma forte restrição para as possíveis soluções e deve, portanto, estar presente na documentação. Mas como iniciar o levantamento?

Quais pontos você deve levantar?

PETERS (2002) sugere o levantamento das seguintes informações:

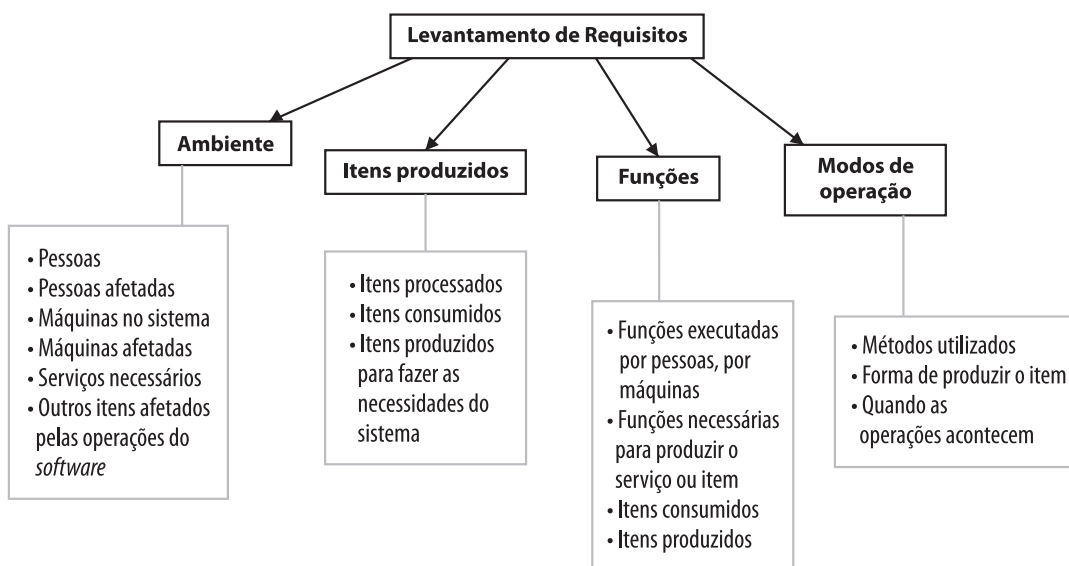



Figura 2.2 - Componentes da Análise do Problema
Fonte: Peters (2000).

Segundo o autor, o levantamento deve ser feito de forma ordenada, dividido em 4 etapas. Primeiro seriam levantados os requisitos relacionados ao ambiente, depois, às funções existentes; ao modo como as funções são executadas e, finalmente, aos itens que são inseridos e suas transformações dentro do processo.

Nunca esqueça: nesta etapa avalie os problemas na situação atual; jamais pense na solução nesta etapa. Concentre-se nas necessidades do cliente.

e) Viabilidade

Antes de você prosseguir é importante considerarmos um estudo da viabilidade do sistema, se vale a pena ou não sua implementação. Para tanto, é necessário que esteja claro se o sistema contribui para com os objetivos da organização, se o sistema pode ser construído usando a tecnologia existente ou ainda se o orçamento comporta o que é necessário para sua implementação.

Um fator forte neste momento de decisões é a possibilidade de integração do sistema aos demais já existentes, se este for o caso. Quando você se certifica da viabilidade está protegendo sua empresa e a empresa do cliente. Este estudo baseia-se nas informações coletadas durante o levantamento de requisitos.

Sommerville (2000) sugere algumas questões para as pessoas da organização, que podem ser colocadas em um cenário de discussão:

- E se o sistema não fosse implementado?
- Quais são os problemas do processo atual?
- Como o sistema proposto irá ajudar?
- Quais serão os problemas de integração?
- São necessárias novas tecnologias? Em quais habilidades?
- Quais recursos devem ser suportados pelo sistema proposto?

Alguns modelos

Em nossa midiateca estão disponíveis alguns modelos para o Levantamento de Requisitos eles podem servir como modelos para futuros levantamentos, dê uma olhadinha:

Roteiro para Análise do Problema – baseado em PETERS(2000)

SEÇÃO 3 - Especificação de requisitos

A sub-etapa de especificação de requisitos visa estabelecer um conjunto de requisitos consistentes e sem ambigüidades, e que possa ser usado como base para o desenvolvimento do *software*.

Nesta etapa também é comum a negociação para resolver conflitos detectados.

> “A especificação de um requisito de *software* é a descrição de um produto de sofware, programa ou conjunto de programa específico que executa uma série de funções do ambiente de destino”. (PADRÃO IEEE 830,1993).

Em resumo, pode-se dizer que **na especificação propomos a solução para o problema do cliente**.

A especificação pode ser totalmente descritiva (como é o caso de alguns documentos da metodologia **RUP** proposta pela *Rational Rose*) ou iniciar a construção de modelos (utilizando-se da notação oferecida pela análise estruturada ou orientada a objetos). Esta decisão depende da metodologia que você estiver utilizando ou mesmo do grau de maturidade da empresa.

RUP - O *Rational Unified Process* é um processo de engenharia de *software* que pretende aumentar a produtividade da equipe oferecendo práticas eficientes executadas por meio de diretrizes, *templates* e orientações sobre ferramentas para todas as atividades críticas de desenvolvimento de *software*.

O uso de uma especificação textual pode ser feito na forma de relatórios. PETERS (2000) sugere um conjunto de quatro questões que devem ser abordadas:

- a primeira delas refere-se às funcionalidades que devem ser definidas e elaboradas, e que o sistema deve suportar;
- a segunda refere-se às interfaces externas do sistema (outros sistemas, equipamentos de *hardware*);
- a terceira relaciona-se ao desempenho esperado pelo produto (questões como tempo de resposta durante uma consulta);
- a quarta relaciona-se a atributos de qualidade (a capacidade do *software* de ser utilizado em diferentes plataformas, por exemplo) e a restrições impostas pelo cliente, pelo ambiente ou pelo próprio desenvolvedor.

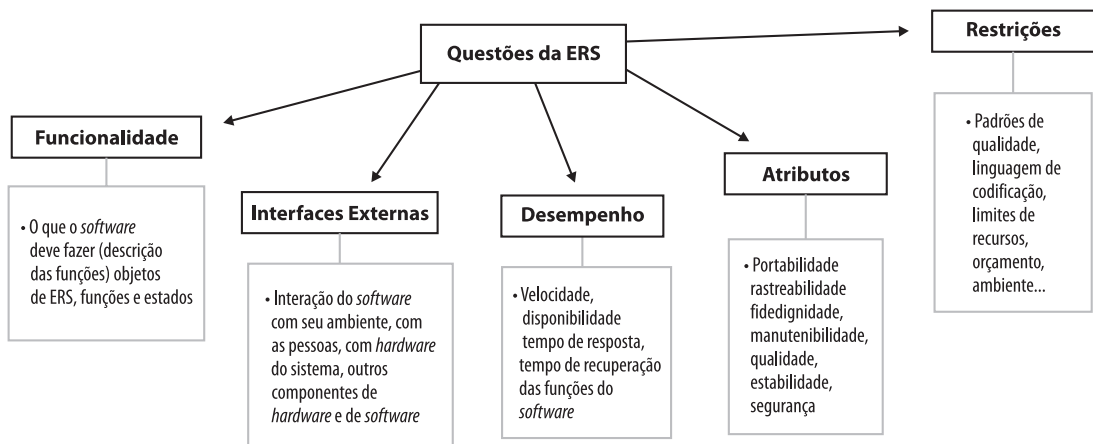



Figura 2.3 - Questões básicas ao se escrever uma ERS
Fonte: Peters (2000).

A especificação dos requisitos faz parte da documentação do sistema, e pode-se dizer que é um de seus principais artefatos. Sabendo disso é necessário decidir "o que" deve ser documentado sobre esta etapa. PETERS (2000) sugere uma estrutura composta de 4 pontos principais :

Introdução (onde são descritas características do documento), descrição global (onde são referenciados perspectivas do produtos, riscos associados e funcionalidades requeridas); requisitos específicos (apresenta um esqueleto de interfaces e necessidades que devem ser suportadas pelo produto em termos de atributos e performance) e a rastreabilidade dos requisitos (que incluem casos de teste para a validação do futuro projeto).

Tabela 2.1 Estrutura para documentação do sistema

1. Introdução	
1.1 Objetivo 1.2 Escopo 1.3 Definições, acrônimos e abreviações 1.4 Referências 1.5 Visão Geral	
2. Descrição Global	
2.1 Perspectiva do produto 2.2 Funções do produto 2.3 Características do Usuário 2.4 Restrições 2.5 Hipóteses e dependências 2.6 Distribuição de requisitos	Perspectiva do produto: seu relacionamento com sistema maior Hipótese: Indicam como alterações feitas na ERS afetam seções específicas da ERS (EX: quais diagramas de fluxo de dados são afetados ou mudanças de funções ou exclusão das mesmas)
3. Requisitos Específicos	
3.1 Interfaces externas, 3.2 Requisitos de processo e dados 3.3 Requisitos de desempenho e qualidade 3.4 Requisitos de banco de dados lógico 3.5 Restrições de projeto 3.6 Atributos de sistema de software 3.7 Organização de requisitos específicos	Restrições de projeto: Políticas regulamentares, limitações de *hardware*, interfaces com outros aplicativos; operação paralela; segurança e perfil, etc.
4. Rastreabilidade dos requisitos	
Apêndice	
Índice remissivo	

Quer conhecer mais ?

Se você estiver interessado neste modelo oferecido por Peters, leia o capítulo 2 do livro:

PETERS, J. F.; PEDRYCZ, W. **Engenharia de** *software*: teoria e prática. Rio de Janeiro: Campus, 2001.

Modelos

A especificação também pode ser feita na forma de modelos. Mas você sabe o que é um modelo?

Um modelo é uma representação de alguma coisa do mundo real, uma abstração da realidade. Além disso, tem a finalidade de servir como fundamento para o projeto de *software*, facilitando a compreensão do fluxo de dados e de controle, do processamento funcional, da operação comportamental e do conteúdo da informação.

Nas primeiras etapas do processo de desenvolvimento (levantamento de requisitos e análise), o engenheiro de *software* representa o sistema através de modelos conceituais, considerando características do sistema independente do ambiente computacional (*hardware* e *software*), no qual o sistema será implementado nas etapas posteriores.

Na etapa seguinte, as características lógicas (características dependentes de um determinado tipo de sistema computacional) e físicas (características dependentes de um sistema computacional específico) são representadas em novos modelos.

A representação dos modelos lógicos e físicos pode ser feita por meio da Analise Estruturada, Análise Essencial ou Análise Orientada a Objetos. Os diagramas utilizados nestas metodologias apóiam e facilitam o entendimento da solução. Nesta disciplina você vai fazer uso dos modelos Orientados a Objetos.

Visite o EVA, ferramenta Midiateca. Lá você encontrará modelos para a Especificação de Requisitos sem a utilização de modelos de análise totalmente textuais. Leia sobre:

Roteiro para Especificação de Requisitos – baseado em Peters (2000).

Software Requirements Specification (without Use-Case) da metodologia RUP - Rational Unified Process

É importante salientar: Ao finalizar a especificação, solicite a assinatura do cliente. Essa assinatura faz com que sua especificação valha como um contrato.

SEÇÃO 4 - Atividades de apoio

As atividades de documentação, verificação, validação e gerenciamento não são exclusivas da etapa de análise de requisitos às atividades de apoio, mas estarão presentes em todo o processo de desenvolvimento do *software*. Dentro do processo de análise de requisitos estas atividades não ocorrem de forma obrigatoriamente seqüencial, mas sim de forma paralela.

a) Documentação de requisitos

É a atividade de representar os resultados da Engenharia de Requisitos em um documento, contendo os requisitos do *software*.

b) Verificação e validação de requisitos

Verificar e validar os requisitos é uma etapa do processo que não pode ser menosprezada. Na verificação de requisitos você avalia se a Especificação de Requisitos está sendo construída de forma correta, seguindo os padrões previamente definidos, evitando requisitos confusos, duplicados, incoerentes, ou incompletos.

A validação verifica se os requisitos e modelos documentados são o reflexo das reais necessidades e requisitos do cliente.

Se você passar por esta etapa em seu projeto, vai evitar a descoberta de interpretações errôneas ou mesmo deficiências do projeto em etapas futuras. Isto significa uma economia de tempo e dinheiro significativa.

Mas, o que você deve observar na validação?

SOMMERVILLE (2000) sugere um check-list, observe:

- O sistema fornece as funções que melhor apóiam as necessidades do usuário?
- Há algum conflito nos requisitos?
- Todas as funções exigidas pelo cliente foram incluídas?
- Os requisitos podem ser implementados com o orçamento e a tecnologia disponíveis?

- O requisito foi apropriadamente compreendido?
- A origem do requisito foi claramente estabelecida?
- O requisito pode ser alterado sem um grande impacto em outros requisitos?

Existem algumas técnicas que apóiam a validação de requisitos:

- As revisões de requisitos pela análise sistemática e regular dos requisitos.
- O uso da prototipação para validar entradas e saídas, por exemplo, por meio de protótipos não-funcionais de telas e relatórios (procure envolver equipe e cliente nessas revisões).
- A geração de casos de teste para os requisitos, verificando se é possível construir os casos de teste e mesmo testar aquele requisito.

c) Gerenciamento de requisitos

A tarefa de gerenciar requisitos se preocupa com as mudanças nos requisitos que já haviam sido acertadas entre cliente e desenvolvedor.

Em outras palavras, é preciso: documentar estas mudanças, gerenciar os relacionamentos entre os requisitos e suas dependências que podem afetar outros requisitos e outros artefatos, produzidos no processo de *software*; verificar a credibilidade da solicitação de mudança com a empresa.

O gerenciamento deve manter as alterações de requisitos de forma controlada, tornando as alterações sustentáveis durante o desenvolvimento.

Fazer mudanças nos requisitos não deve ser uma catástrofe, é comum a dificuldade de reconhecer todos eles em uma etapa inicial. O importante é documentar, relacionar, controlar estas mudanças, repassando-as a todo o processo e envolvidos no processo de desenvolvimento.

Agora, para praticar os conhecimentos conquistados nesta unidade, realize as atividades propostas.

Atividades de auto-avaliação

Leia com atenção os enunciados e, após, realize as questões propostas:

1) Quanto ao requisito, é correto afirmar:

a) ( ) Um requisito é expresso por suas características funcionais.

b) ( ) O requisito de *software* pode ser expresso por características implícitas, explícitas e normativas.

c) ( ) As características normativas de um requisito estão relacionadas as suas características comportamentais.

d) ( ) O requisito mais importante em uma análise é o requisito explícito.

2) Na análise de requisitos, temos as seguintes etapas:

(a) levantamento

(b) especificação

(c) validação

(d) gerência

Relacione as descrições abaixo com a etapa correspondente.

a) ( ) responsável pelo controle de alterações de requisitos;

b) ( ) utiliza-se de técnicas, como observação direta, entrevistas e questionários para apurar informações pertinentes ao processo;

c) ( ) avalia se a Especificação de Requisitos está seguindo os padrões previamente definidos, evitando requisitos confusos, duplicados; incoerentes; ou incompletos;

d) ( ) responsável pela definição de restrições, funcionalidades; atributos; interfaces externas e desempenho.

3) No texto abaixo voce tem a solicitação para desenvolvimento de *software* de um cliente proprietário de uma clinica médica. A partir do texto levantado junto ao cliente, preencha o documento de análise do problema que dará inicio a documentação do futuro sistema.

Empresa : Clínica Bem-Estar

1) Função: fomos contratados para analisar seu processo atual e verificar como expandir suas operações e melhorar seu nível de serviço.

Histórico:

A clínica fundada há 5 anos atua no atendimento clinico-pediátrico.

A clínica possui 34 médicos cadastrados em diferentes especialidades como: cardiologia; clínica geral; dermatologia, etc. Todos os médicos utilizam internet e e-mail. A faixa etária predominante é de 30, 35, 40, 42, 44 e 48 anos. Todos os médicos são aptos sob o ponto de vista físico.

O paciente pode ser atendido de forma particular ou por convênios. Os convênios atendidos são o Bruxtr, Vpfzm e UIOlk.

Cada médico faz 3 plantões semanais de 4 horas seguidas; as consultas possuem um intervalo de 30 minutos. Existe a possibilidade da consulta ser de retorno, neste caso são apenas 15 minutos.

A clínica é 24 horas. Cada médico possui uma agenda preta onde são marcadas as consultas. Na marcação da consulta é colocado o nome do paciente, horário e convênio. Trabalham à 3 anos na clínica com planilhas excel.

A clínica possui 2 atendentes que são responsáveis por preencher o cadastro inicial do paciente que contém nome; endereço; telefone; data de nascimento; convênio.

O médico, ao atender o paciente, preenche sua ficha manualmente, informando peso, altura, idade, motivo da consulta, queixa principal, doenças anteriores, diagnóstico, prescrição. A prescrição pode ser a solicitação de exames ou medicamentos com posologia.

A clínica possui aproximadamente de 700 à 800 fichas, sendo que entorno de 600 são atendidas por convênio.

O gerente da clínica está ansioso, pois não consegue controlar questões relacionadas ao número de pacientes atendidos por convênio e particular; médicos mais procurados e picos de movimento.

Volume de atendimentos: 56 por dia.

Outra questão de interesse é manter um controle de laboratórios conveniados, pois o médico poderia indicar o laboratório já no momento da prescrição.

TEMPLATE PARA REALIZAÇÃO DA ANÁLISE DO PROBLEMA CLINICA BEM-ESTAR

1 - Nome da empresa: ____

2 - Contato: ____

3 - Descrição do problema

____

____

____

____

____

4 - Identificação do principal objetivo do cliente.

____

____

5 - Descrição dos usuários do sistema (para cada usuário descreva cargo e possíveis funções dentro do processo).

____

____

____

____

____

6 - Descrição detalhada dos processos existentes (COMO O SISTEMA ATUAL FUNCIONA?)

Para marcação da consulta:

____

____

Síntese

O objetivo do desenvolvedor de *software* sempre deve ser o fornecimento de um *software* de qualidade que atenda as necessidades dos clientes. Nesta unidade você aprendeu que para obter essa qualidade, é preciso realizar a etapa de análise de requisitos de forma consistente, utilizando-se metodologias apropriadas que permitam a revisão constante de todo o processo. O uso de diferentes técnicas, como entrevistas, questionários durante o levantamento de requisitos ajudam a variar o foco da observação do projetista. Na especificação o uso de diferentes modelos pode ser adaptado de acordo com as características da empresa. As atividades de apoio, como o gerenciamento de requisitos, permitem rastrear alterações de requisitos e comunicar estas alterações a toda equipe envolvida.

A análise de requisitos é um momento de conhecimento, reconhecimento e de explosão de idéias. O bom aproveitamento desta etapa é fundamental; estudos demonstram que o custo de alterações em etapas posteriores à análise de requisitos podem chegar a 100% do custo original do projeto.

Nesta unidade você também percebeu que a análise de requisitos incorpora o uso de modelos; o uso de modelos permite desenvolver o *software* de maneira previsível em um determinado período, com utilização eficiente e eficaz de recursos; o modelo ajuda a visualizar o sistema como desejamos simplificando seu entendimento.

Na próxima unidado você vai dar início ao estudo sobre a Análise Estruturada e suas notações, permitindo que você complemente a análise de requisitos, utilizando-se da notação na formação dos modelos.

Saiba mais

Para aprofundar as questões abordadas nesta unidade você poderá pesquise em:

- SOMMERVILLE, Ian. **Engenharia de** *software*. 6 ed. São Paulo: Addison-Wesley, 2003.

UNIDADE 3

Análise Estruturada

Objetivos de aprendizagem

- Reconhecer objetivos e características inerentes ao uso da modelagem estruturada.
- Fazer uso de conceitos e diagramas da modelagem estruturada.
- Compreender e reconhecer uma estrutura que se utilize da modelagem estruturada.
- Empreender o uso da modelagem estruturada.

Seções de estudo

Seção 1 Análise estruturada

Seção 2 Diagrama de Fluxo de Dados (DFD)

Seção 3 Dicionário de Dados

Seção 4 Modelagem de Dados

Para início de estudo

Quando se inicia o processo de desenvolvimento de um *software*, são realizadas tarefas específicas que podem ser estruturadas dentro de um modelo de desenvolvimento (como você estudou na primeira unidade da disciplina).

No início do processo tem-se a etapa de análise de requisitos (assunto estudado na segunda unidade), na qual você é confrontado com as reais necessidades de seu cliente.

Ainda que você esteja ansioso por começar o processo de desenvolvimento e efetivamente iniciar a etapa de codificação, a etapa crucial do processo ainda está por vir: o projeto de desenvolvimento de *software*.

Como você estudou na primeira unidade, o projeto está contido na etapa de desenvolvimento do *software*, e é nela que você vai desenhar o modelo da solução que deve resolver as necessidades apontadas no levantamento de requisitos. Existem diferentes metodologias para cumprir esta etapa, mas as mais conhecidas são a Análise estruturada e a Análise orientada a objetos. Na análise estruturada, o desenvolvimento do sistema é voltado para as funções (processos) que ele deve realizar. Durante a unidade 3, você vai conhecer as notações e características específicas utilizadas em uma modelagem estruturada e que são fundamentais em sua documentação.

SEÇÃO 1 - Análise estruturada

A Análise estruturada foi desenvolvida nos anos 70 por Gane, Sarson e De Marco. A técnica é baseada na utilização de uma linguagem gráfica para construir modelos de um sistema, incorporando também conceitos relacionados às estruturas de dados.

YOURDAN (1992) esclarece algumas características da análise estruturada:

Quando falamos em análise estruturada temos que ter em mente que todo o desenvolvimento do sistema é voltado para as funções que o sistema deve realizar.

O que são funções? Imagine o sistema de uma vídeo locadora, nele você tem cadastro de clientes, cadastro de fitas, relatório de clientes da locadora entre outras funções que o sistema deve executar. As funções utilizam informações e produzem informações para o ambiente (na função cadastro de fita o atendente faz a entrada de dados no cadastro de fita digitando informações sobre a fita, como nome do filme, atores, tipo de filme, etc). Estas informações são repassadas às entidades externas (que pode ser o cliente, o atendente da locadora) por meio de fluxos de dados ou armazenadas em depósitos de dados (são os depósitos que vão armazenar em seu HD as informações do cliente, das fitas, etc).

Retomando o exercício da Clínica Bem-Estar.

Empresa : Clínica Bem-Estar

1) Função: fomos contratados para analisar seu processo atual e verificar como expandir suas operações e melhorar seu nível de serviço.

Histórico:

A clínica fundada há 5 anos atua no atendimento clinico-pediátrico.

A clínica possui 34 médicos cadastrados em diferentes especialidades como: cardiologia; clínica geral; dermatologia, etc. Todos os médicos utilizam internet e e-mail. A faixa etária predominante é de 30, 35, 40, 42, 44 e 48 anos. Todos os médicos são aptos sob o ponto de vista físico.

O paciente pode ser atendido de forma particular ou por convênios. Os convênios atendidos são o Bruxtr, Vpfzm e UIOlk.

Cada médico faz 3 plantões semanais de 4 horas seguidas; as consultas possuem um intervalo de 30 minutos. Existe a possibilidade da consulta ser de retorno, neste caso são apenas 15 minutos.

A clínica é 24 horas. Cada médico possui uma agenda preta onde são marcadas as consultas. Na marcação da consulta é colocado o nome do paciente, horário e convênio. Trabalham à 3 anos na clínica com planilhas excel.

A clínica possui 2 atendentes que são responsáveis por preencher o cadastro inicial do paciente que contém nome; endereço; telefone; data de nascimento; convênio.

O médico, ao atender o paciente, preenche sua ficha manualmente, informando peso, altura, idade, motivo da consulta, queixa principal, doenças anteriores, diagnóstico, prescrição. A prescrição pode ser a solicitação de exames ou medicamentos com posologia.

A clínica possui aproximadamente de 700 à 800 fichas, sendo que entorno de 600 são atendidas por convênio.

O gerente da clínica está ansioso, pois não consegue controlar questões relacionadas ao número de pacientes atendidos por convênio e particular; médicos mais procurados e picos de movimento.

Volume de atendimentos: 56 por dia.

Outra questão de interesse é manter um controle de laboratórios conveniados, pois o médico poderia indicar o laboratório já no momento da prescrição.

Se você fosse listar alguns dos requisitos funcionais com certeza poderia listar:

- Cadastro de médicos que possibilite cadastrar, alterar e excluir os dados cadastrais dos médicos da clínica.
- Cadastro de pacientes que permita cadastrar, alterar e excluir os dados cadastrais dos pacientes da clínica.
- Agenda médica onde deve ser possível o agendamento, consulta e exclusão do agendamento de consulta de um paciente, com um determinado médico, em um determinado horário.
- Ficha médica que permita o lançamento e consulta de dados sobre a consulta do paciente.
- Emissão de relatório estatístico sobre o atendimento de convênios.
- Emissão de relatório estatístico sobre o atendimento por médicos.
- Emissão de relatório estatístico sobre o número de atendimentos durante o ano .
- Cadastro de laboratórios conveniados à clínica.

Se fossemos listar os requisitos não funcionais:

- O sistema deve operar de forma confiável no lançamento de todos os dados de pacientes e consultas e sua gravação.
- O sistema deve proteger o acesso às informações históricas da ficha do paciente, sendo que seu acesso deve ser exclusivo para a equipe médica
- O sistema deve ser desenvolvido sobre uma plataforma *open source* permitindo portabilidade.
- O sistema deve seguir requisitos de usabilidade para as interfaces, sendo de fácil aprendizado pois a rotatividade das atendentes é grande.

A partir da lista de requisitos você pode iniciar o processo de modelagem. Para tanto é necessário entender alguns conceitos básicos.

A notação da análise estruturada é composta pelos seguintes elementos:

- Diagrama de Fluxo de Dados (DFD),
- Dicionário de dados;
- Especificação dos processos; e,
- Modelagem de dados (ER).

Nas próximas seções, você vai conhecer detalhadamente cada um desses elementos.

SEÇÃO 2 - Diagrama de Fluxo de Dados (DFD)

Segundo FORTES (1999), um Diagrama de Fluxo de Dados (DFD) é uma técnica gráfica de representação que permite explicitar os fluxos de informação e as transformações que são aplicadas, à medida que os dados se deslocam da entrada em direção à saída.

O DFD permite particionar o sistema, demonstrando seus componentes ativos (como usuários, equipamentos, arquivos) e o interfacemento de dados que ocorre entre eles.

Para realizar um DFD, são necessários alguns símbolos que o compõem. Conheça cada um:

Processo	■ A bolha ou círculo que representa o **processo**. O processo é um executor de tarefas – funções, atividades. Representa tarefas de processamento do sistema. ■ Quando você colocar nome em um processo, procure utilizar um nome que descreva **o que** o processo faz. ■ Um exemplo de processo pode ser: Cadastrar paciente, Cadastrar Médico, Agendar Consulta
Entidades externas	■ O retângulo representa **entidades externas (EE)**. A entidade externa representa origens e destinos dos dados que percorrem o sistema. Quando você pensar em entidade externa pense que são criadores e/ou consumidores de dados/informação. ■ Uma EE pode ser uma pessoa, organização ou outros sistemas que interagem com o sistema para envio e/ou recebimento de informações. Pode representar a interface entre o sistema e o mundo externo. ■ A EE pode ser então, no caso da Clinica Médica, o Paciente, o Atendente, o Médico. Se na clínica houver um laboratório que de alguma maneira receba dados da clínica por meio do sistema, então, neste caso o Laboratório pode ser considerado uma EE.
Depósito de dados	■ A caixa aberta, ou linha dupla preenchida com um rótulo, representa o **depósito de dados**. Na verdade são os locais onde ocorre o armazenamento de dados, é um repositório de dados (temporários ou permanentes). ■ Todas as informações do paciente como nome, endereço, telefone serão armazenados no seu HD em um deposito de dados, neste exemplo um nome adequado para o depósito seria Paciente.
Fluxo de dados	■ O **fluxo de dados** é expresso por setas rotuladas que interligam os processos e que permitem indicar o caminho seguido pelos dados. O fluxo de dados realiza a comunicação entre os processos, depósitos, e entidades externas. ■ O fluxo de dados não significa seqüência, de um módulo de programa para outro. Mas, a seta indica o caminho pelo qual uma ou mais estruturas de dados deverão passar. ■ Quando estamos realizando o processo cadastrar paciente, deve existir um fluxo de dados, em que são repassadas do paciente ao processo cadastrar paciente, as suas informações cadastrais (nome, endereço...).

Acompanhe na figura 3.1, a seguir, o processo Cadastrar Paciente.

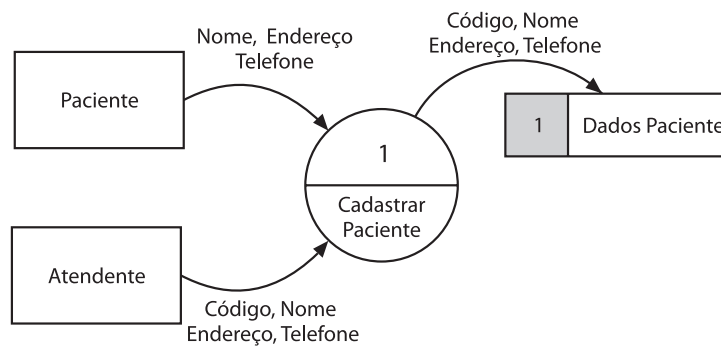



Figura 3.1 - DFD (processo Cadastrar Paciente)

São possíveis perguntas que você pode realizar:

- Por que Cadastrar Paciente (círculo) está representado como um processo?

Resposta: Ele é um processo porque representa uma execução, uma função que deve ser executada no sistema para que o usuário alcance seu objetivo que é fazer o cadastro do paciente para a clínica.

- Quem são as entidades externas (retângulo) para este processo?

Resposta: São todos os consumidores e criadores das informações que este processo irá gerar. Observe que para cadastrar um paciente é necessário que o paciente informe seus dados, além dele temos a atendente que vai inserir os dados do paciente no sistema. Assim este processo tem no mínimo duas entidades externas: Paciente e Atendente.

- Por que Dados Paciente é um depósito de dados no diagrama?

Resposta: Todas as informações do paciente devem ser armazenadas no sistema (se não fosse assim o paciente teria que informar seus dados a cada consulta na clinica).

Para representar a armazenagem dos dados a notação usada é o depósito de dados. Portanto neste processo (Figura 3.1) os dados do paciente representam um depósito de dados onde as informações do paciente serão "guardadas" pelo sistema na clínica.

Seria assim: a entidade externa informa seus dados para o processo Cadastrar Paciente (nome, endereço, telefone), a Atendente (entidade externa), por sua vez, informa os dados ao processo cadastrar paciente (por meio do fluxo de dados).

Após alimentarem o processo, essas informações são transformadas e os Dados do Paciente são armazenados no depósito de Dados Paciente.

Observe a figura número 1 no processo Cadastrar Paciente, ou o número 1 no depósito de Dados Paciente. Estes números são utilizados apenas para organizar o DFD para facilitar sua leitura. Na verdade são seqüenciais e sua utilização é opcional.

Na documentação você pode se referir ao processo por seu número não sendo necessário escrever todo o nome do processo.

Quando se elabora um DFD, faz-se necessária a observação de algumas diretrizes.

Lembre-se de que quando usamos Fluxos de dados, estamos representando o deslocamento de informações, e isto pode acontecer **somente**:

- entre um Processo e uma Entidade Externa;
- entre dois Processos;
- entre um Processo e um Depósito de Dados.

Quando você nomear o processo, procure utilizar verbos no infinitivo. As Entidades Externas, Fluxos de Dados e Depósitos devem ser identificados por substantivos.

Existe um fator muito importante em um DFD: ele não é um fluxograma, portanto ele jamais deve representar a seqüência em que os processos são ativados.

O DFD que é apresentado na figura 3.2 é bastante genérico e pode deixar margens a dúvidas. Quando isto acontece, é possível "explodir" o DFD em níveis de profundidade diferente. A decisão do número de níveis necessários depende da complexidade do projeto: quanto maior o número de níveis, melhor a descrição do comportamento do sistema.

Os níveis podem ser descritos como:

a) Nível 0 (Topo): também chamado de **Diagrama de Contexto.** É formado por um único processo que representa o sistema inteiro.

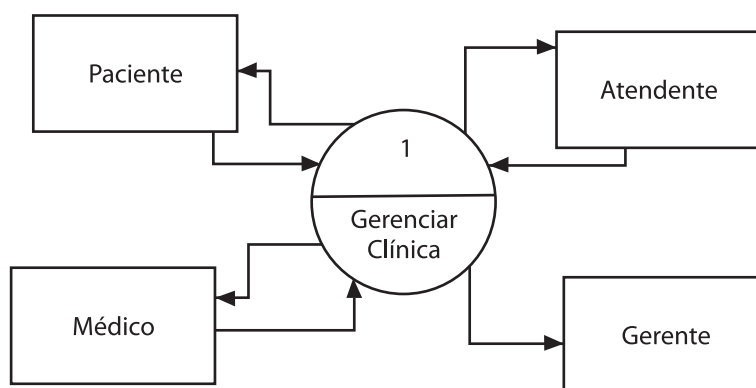



Figura 3.2 - DFD - Nível 0

b) Nível n (Folhas): neste caso, os processos são primitivos, sua descrição fica em um nível alto, pois as funções são simples, com poucas E/S.

c) Nível 1 a n-1 (Intermediários): neste caso, os processos são decompostos em níveis. O **n+i** é a decomposição/particionamento de processos, fluxos e depósitos do nível **i**. Em outras palavras: você pode decompor o DFD de forma granular, chegando a um detalhamento minucioso.

Veja o exemplo do DFD Agendar Consulta. Ele pode ser apresentado em um nível apenas:

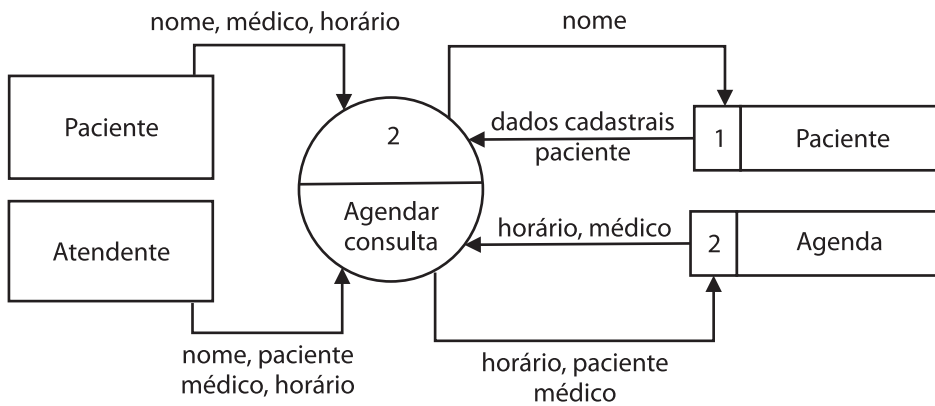



Figura 3.3 - Agenda Consulta Nível 1

Mas, se você quiser detalhar estes processos o DFD oferece mais informações para o desenvolvedor:

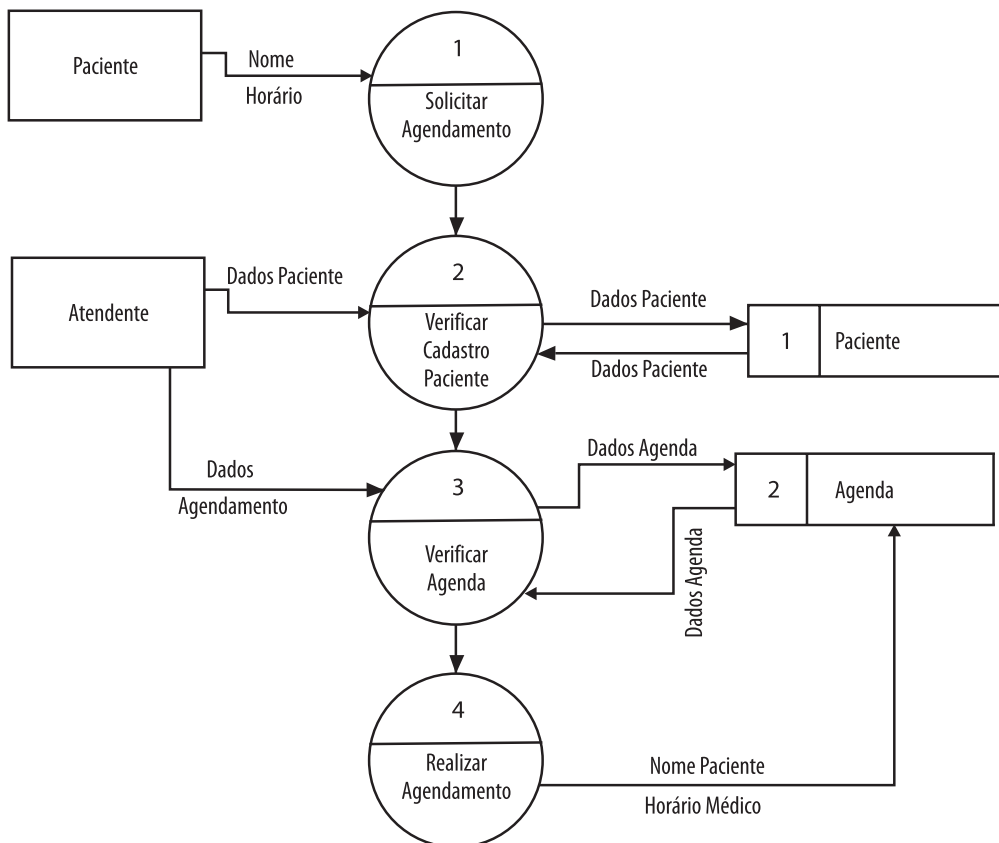



Figura 3.4 - DFD - Agendar Consulta

A figura 3.4 é um exemplo da explosão do DFD Agendar Consulta. Agora já é possível perceber que existem pelo menos 4 subprocessos envolvidos no agendamento da consulta:

- solicitar agendamento
(disparado pela Entidade Exterma - EE paciente).
- verificar cadastro paciente (onde será verificada a existência ou não de cadastro do paciente na clínica médica).
- verificar agenda (nela será verificado o horário solicitado, o médico e suas disponibilidades para a consulta).
- realizar agendamento (processo que efetiva a marcação da consulta, armazenando a informação no depósito de dados).

Dicas para a construção de DFD´s

Os DFD's são construídos a partir da lista de funcionalidades propostas para o sistema.

a) Escolha nomes significativos para os processos, fluxos, depósitos e entidades externas. Os nomes devem ser facilmente identificados, principalmente pelo vocabulário do usuário.

b) Numere os processos. Em um sistema grande a numeração pode ser muito útil.

C) Sempre tenha em mente a facilidade de entendimento do DFD. Se o DFD estiver confuso, refaça-o até que lhe pareça suficientemente claro.

A figura 3.5 apresenta um DFD que representa os processos a serem realizados para o controle e faturamento de um sistema de pedidos.

Observe que neste caso temos 3 processos envolvidos para a realização do pedido. Dois depósitos de dados (Faturas e Pedidos) e apenas uma entidade externa (Clientes). A comunicação entre processos evolui pela passagem de dados por meio dos fluxos de dados.

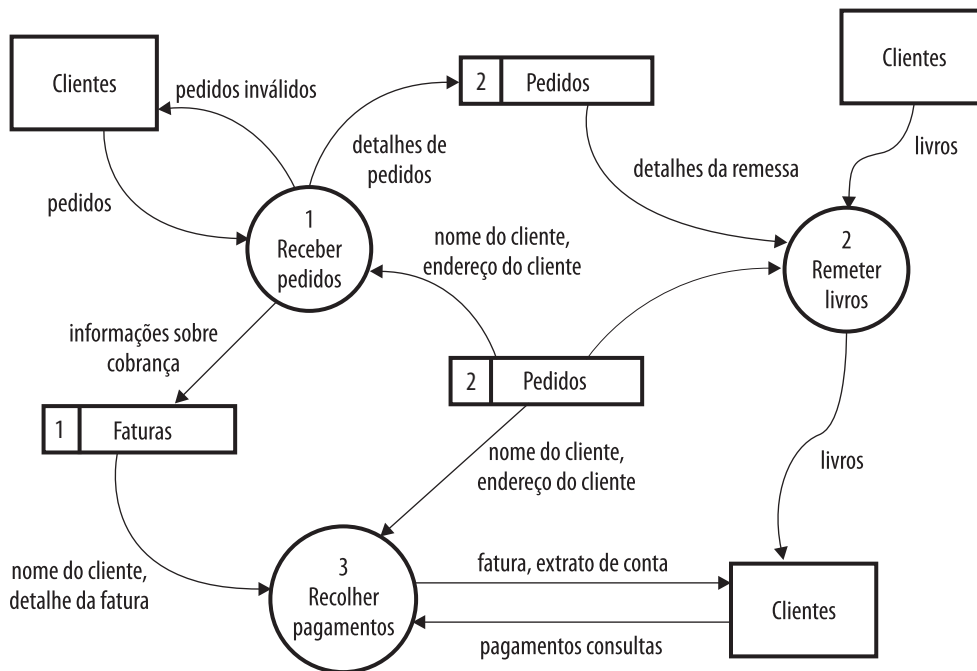



Figura 3.5 - DFD Controle de Pedidos (Yourdon, 1992)

Observe a seguinte situação: você recebe a missão de desenvolver a modelagem estruturada de um sistema onde se pretende o desenvolvimento de um controle de pedidos para uma floricultura atacadista. A empresa, em questão, faz a revenda de flores e folhagens para floriculturas da cidade não tendo venda a varejo.

Os requisitos funcionais da floricultura são:

- Manter o cadastro de produtos a venda na floricultura (a venda é exclusivamente de flores e folhagens)

- Manter o cadastro de clientes da floricultura
- Manter o controle de pedidos da floricultura
- Controlar o pagamento de pedidos realizado por clientes da floricultura.

Os requisitos não funcionais da floricultura são:

- São requisitos não funcionais a manutenção da simplicidade do código permitindo a facilidade na manutenção.

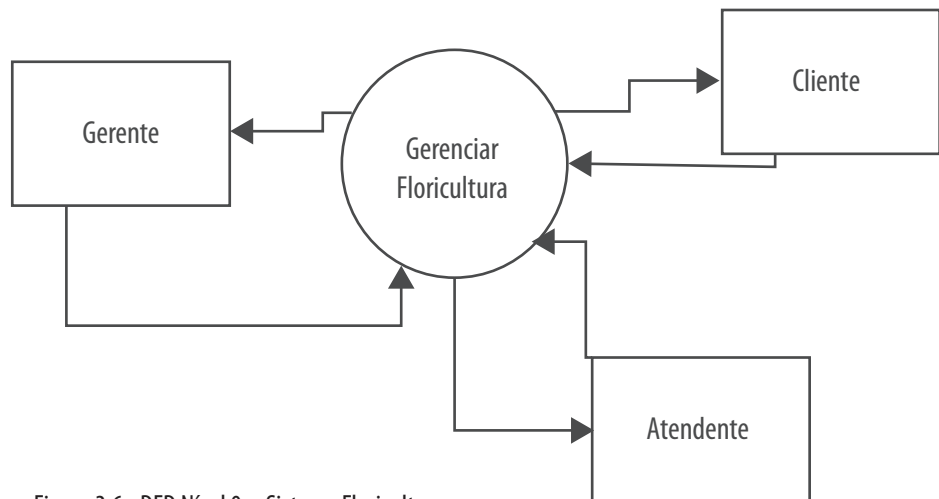



Figura 3.6 - DFD Nível 0 – Sistema Floricultura

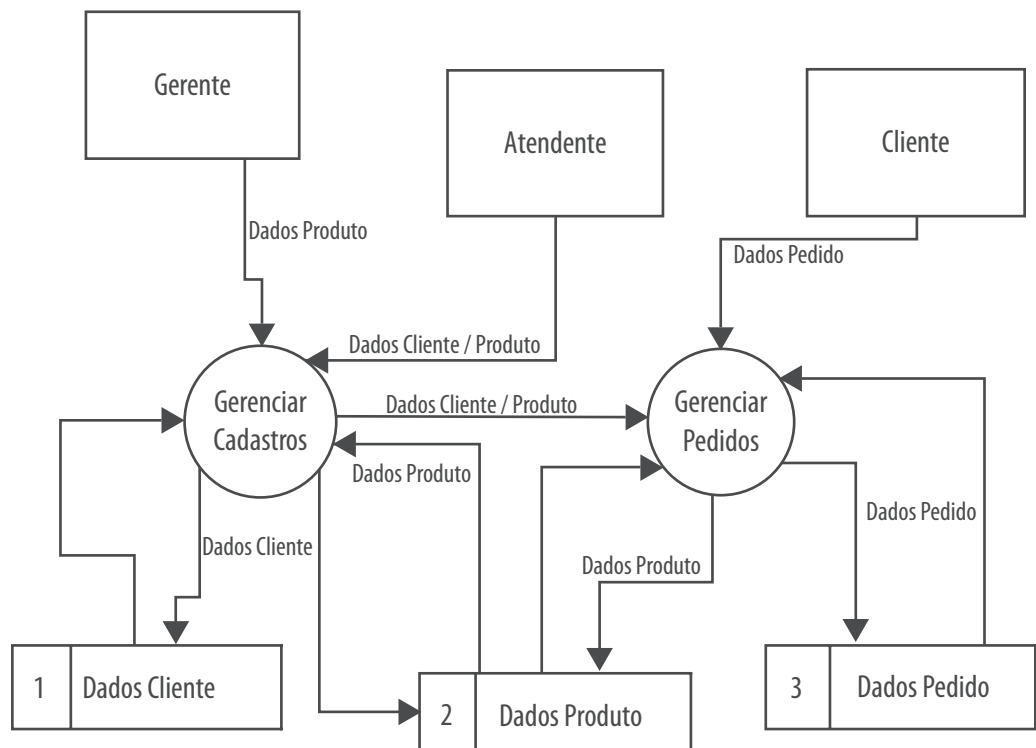



Figura 3.7 - DFD Nível 01 – Sistema Floricultura

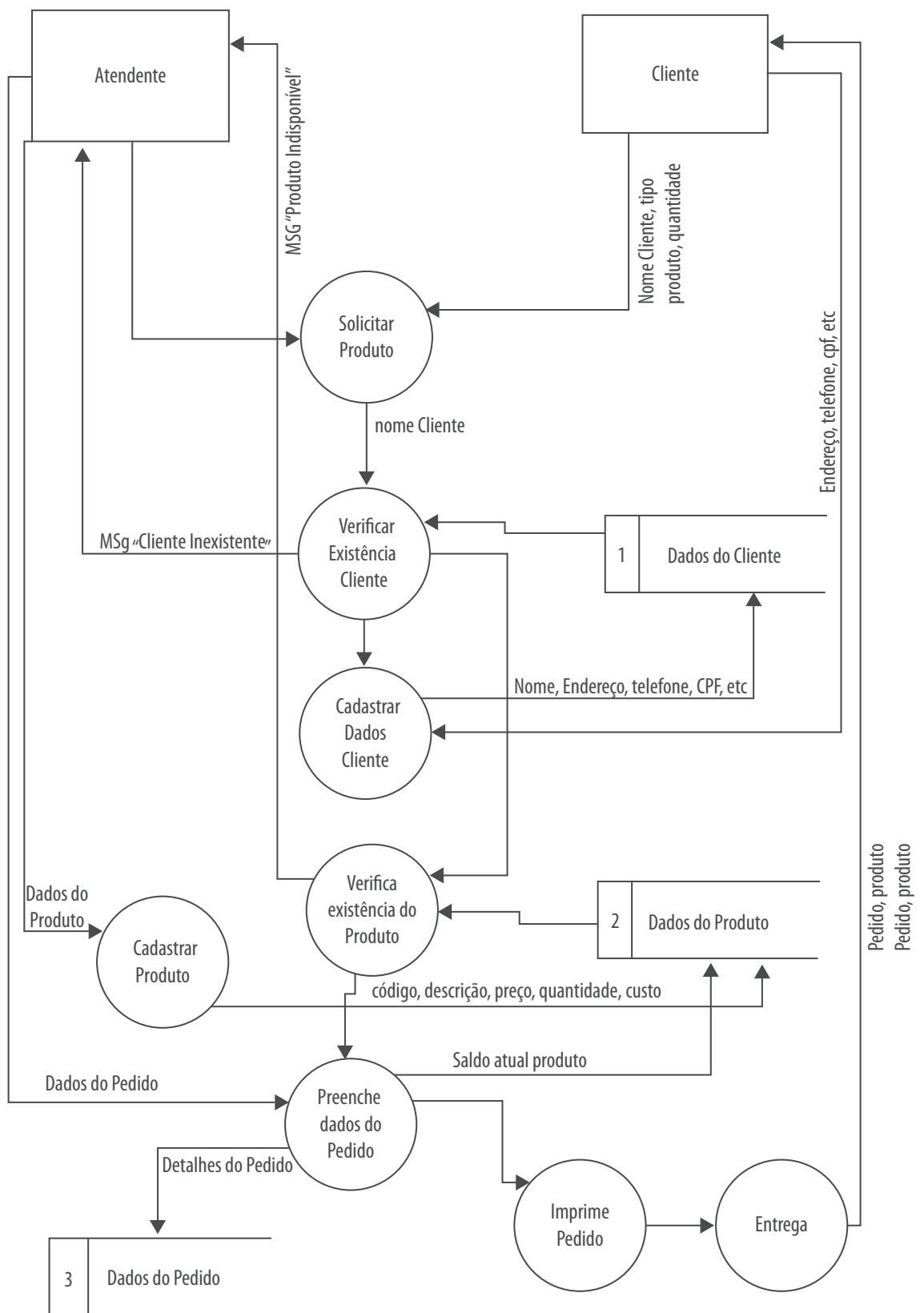



Figura 3.8 - DFD Nível 2 – Sistema Floricultura

SEÇÃO 3 - Dicionário de Dados

O dicionário de dados é uma descrição precisa de todas as informações presentes no DFD para que usuários e analistas possam compreender de forma simplificada todo o modelo.

Quando se constrói um DFD, os componentes são identificados por um rótulo único. Na figura 4, no exemplo da clínica, você tem o depósito de dados Agenda.

Apenas o rótulo apresentado no DFD não define a estrutura dos dados que irão representar a informação. Essa definição é feita no dicionário de dados.

Mas, o que deve ser descrito no depósito de dados?

Pode-se definir o significado de depósitos, a descrição dos fluxos de dados, a estrutura dos arquivos e a especificação lógica dos processos.

As notações mais diversas podem ser utilizadas para definir as estruturas dos dados, sendo que uma proposta de notação possível, de acordo com MAZZOLA (1999), inclui as seguintes informações:

- **nome**, o identificador principal do item de dados, do depósito de dados ou de uma entidade externa; e, eventualmente, outros nomes utilizados para o mesmo item;
- **especificação** (numérico, alfanumérico, data, inteiro, tamanho máximo, formatação, domínio);
- **utilização**, em que contexto (onde e como) o item de informação é utilizado;
- **descrição**, uma notação que permita explicitar o conteúdo do item;
- **informações complementares** a respeito do item de dados, como valores iniciais, restrições, etc.

Observe o exemplo de um dicionário de dados para o depósito de dados Paciente (figura 3.9).

Depósito

Nome do Depósito: Paciente
Especificação : Banco de dados cadastrais do paciente, volume aproximado 3500 registros.
Descrição: o depósito de dados Paciente deve armazenar todos os dados cadastrais do paciente da clínica tendo como chave o nome do paciente.
Utilização : o depósito será usado no processo Cadastrar Paciente, Agendar Consulta.

Atributos do Depósito de Dados Paciente:

Nome do Atributo: Nome_Paciente
Especificação : Campo alfanumérico, tamanho 50, chave primária do depósito de dados.
Descrição: Nome do paciente .

Nome do Atributo: Endereço_Paciente
Especificação : Campo alfanumérico, tamanho 60 caracteres.
Descrição: Endereço do paciente .

Nome do Atributo: Telefone_Paciente
Especificação : Campo alfanumérico, tamanho 12.
Descrição: Telefone do paciente .

Nome do Atributo: Sexo_Paciente
Especificação : Campo alfanumérico, tamanho 1, Assume os valores "F" ou "M"
Descrição: Sexo do paciente.

Figura 3.9 Exemplos de dicionário de dados para o depósito de dados do Paciente.

SEÇÃO 4 - Modelagem de Dados

A modelagem de dados é também conhecida como Diagrama E-R (Entidade -Relacionamento). Esta técnica foi desenvolvida originalmente para dar suporte ao projeto de bancos de dados CHEN (1990).

O modelo ER é uma técnica utilizada para representar os dados a serem armazenados em um sistema.

A simbologia utilizada em um diagrama ER é composta pelos seguintes elementos:

paciente Entidade	Uma entidade representa um objeto concreto ou abstrato onde serão armazenadas as informações. Uma entidade pode ser uma pessoa, uma instituição, elementos do domínio da aplicação (Paciente, Agenda). Quando você cria uma entidade, a entidade é composta por um conjunto de instâncias. Cada instância é uma única ocorrência de uma determinada entidade. Em um depósito Paciente, João Dirceu é uma instância de Paciente.
Atributo	Os atributos são utilizados para descreverem características ou propriedades elementares de entidades e relacionamentos. Os atributos representam o conteúdo de uma entidade. Pensando no exemplo da clínica, alguns atributos são NomePaciente, TelefonePaciente, SexoPaciente, etc.
Relacionamento	Um relacionamento é uma abstração de uma associação entre duas ou mais entidades. Pode haver mais de um relacionamento entre objetos. Um exemplo de relacionamento ocorre quando pensamos que o paciente Almir (uma instância da entidade Paciente) possui “n” agendamentos (na entidade Agenda).

- Imagine o sistema da Clínica. Quais entidades você consegue identificar?

Com certeza é necessário armazenar os dados do Paciente, os dados e informações do Médico, os dados para o agendamento da consulta, o Agenda_Consultas e os dados da Consulta (informações do médico durante a consulta).

Observe, você identificou assim grandes grupos de informação que necessariamente precisam ser armazenados para uso futuro na forma de consultas e relatórios (que são processos que irão atuar sobre os dados).

A entidade Paciente teria os seguintes atributos.

Nome da entidade: Paciente

Chave identificadora: Nome Paciente

Atributos da entidade:
Cpf
Telefone
Endereço
Convênio
data de nascimento

Exemplos de instâncias do objeto Paciente seriam:

nome: Paulo Lopes	nome: Joana da Silva
telefone: 88890899	telefone: 8887777
cpf: 483.432.546-65	cpf: 488.444.449-35
endereço: Rua do Pântano 120	endereço: Rua do Rio 26

Na figura a seguir, você pode ver um recorte de um diagrama ER. Observe que é a relação entre a entidade paciente que realiza um agendamento na clínica.

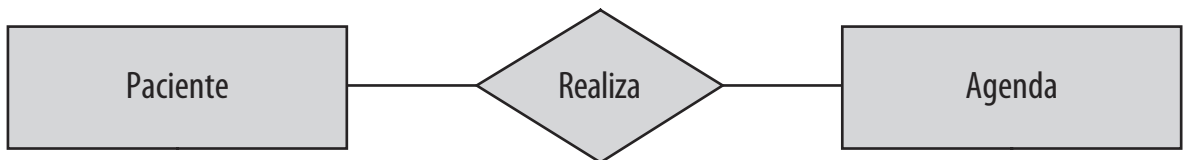



Figura 3.10 - Recorte de um diagrama ER

Se você pensar na entidade Paciente, existente na clínica, pode representar seus atributos por meio da notação gráfica. O atributo Nome é sublinhado porque considera-se que ele é uma chave nesta entidade.Cada um dos círculos representa um atributo.

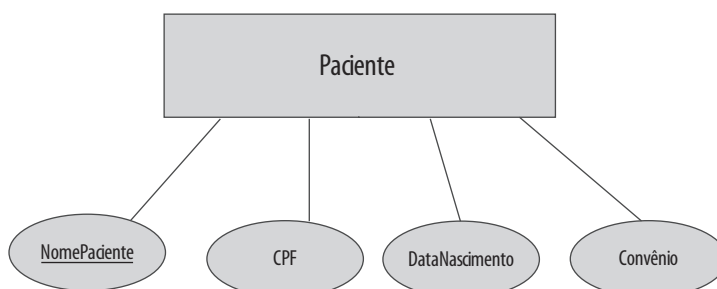



Figura 3.11 – Notação gráfica entidade Paciente

Observe os exemplos a seguir. São exemplos de relacionamento, porém, sem a indicação do número de relacionamentos existentes entre eles.

Figura 3.12 – Exemplos de relacionamento

Observe na figura 3.12, que você está lendo as entidades e o relacionamento entre elas:

Cliente (entidade que armazena os dados de um cliente de vídeo locadora) **Aloca Filme** (entidade que armazena os dados de filmes na locadora)

Médico (entidade que armazena os dados dos médicos na clínica) **Atende Paciente** (entidade que armazena os dados de pacientes na clínica)

Pedidos (entidade que armazena os dados de um pedido de compras) **Contém Produtos** (entidade que armazena os dados dos produtos)

Cliente (entidade que armazena os dados de um cliente de um banco) **Tem Cartão** (entidade que armazena os dados da conta e do cartão que o cliente possui do banco)

Quando você identifica quantas vezes cada instância de uma entidade pode participar do relacionamento, você está

determinando a classe desse relacionamento. A cardinalidade indica o número máximo e mínimo de associações possíveis em um relacionamento.

Você pode ter classes de:

a) Cardinalidade 1 para 1 (1:1)

Quando a cardinalidade é de 1 para 1, significa que cada instância da primeira entidade pode ser relacionada com exatamente uma instância da segunda entidade, ou seja, cada cliente tem um cartão, como no exemplo a seguir, em que Ana Lopes tem um, e apenas um cartão, cujo número é 23456-7, e o cartão tem somente 1 cliente, a Ana Lopes.

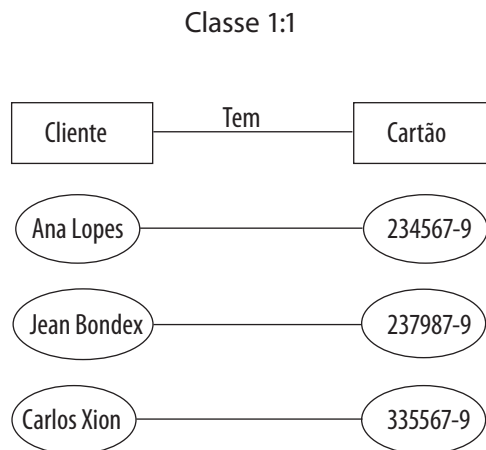



Figura 3.13 - Cardinalidade 1 para 1 (1:1)

b) Cardinalidade 1 para N (1:N)

Se a cardinalidade for 1:N, tem-se então uma situação como a ilustração a seguir, na qual um cliente pode ter de 0 (nenhuma) a “n” fitas alocadas. A instância João tem alocadas 3 fitas (instâncias).

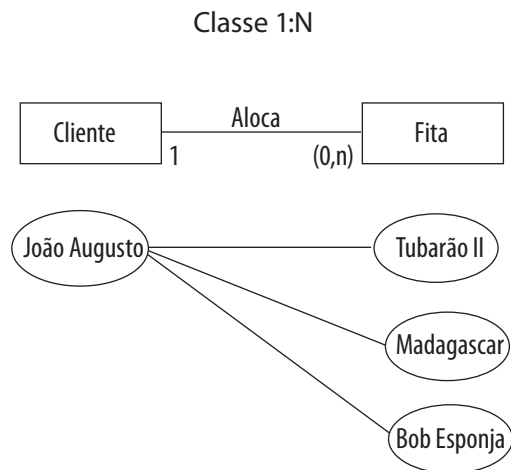



Figura 3.14 - Cardinalidade 1 para N (1:N)

c) Cardinalidade N para N (N:N)

A situação N para N representa uma situação de relacionamento de muitos para muitos. Nesse caso, uma instância da primeira entidade se relaciona com uma ou mais instâncias da segunda entidade. A instância da segunda entidade também estabelece relacionamento com uma ou mais instâncias da primeira entidade. No exemplo a seguir, o funcionário João Augusto pode estar relacionado a diversos projetos. O projeto Folha de pagamento tem alocados vários funcionários.

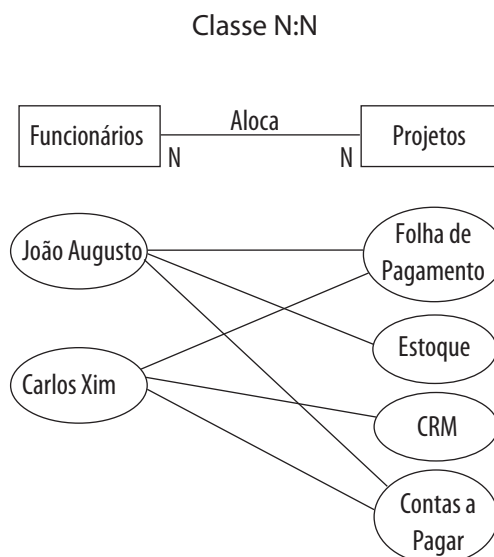



Figura 3.15 - Cardinalidade N para N (N:N)

Observe algumas dicas importantes:

Identifique, primeiro, quais são as entidades do sistema.

- Identifique, então, os atributos para cada entidade.
- Determine as chaves das entidades: não esqueça que a chave **deve** ser um atributo que distinga a instância de forma única em relação as outras instâncias da entidade: Um exemplo disso é o registro do CPF. Cada cidadão possui o seu, é único, não existe nenhum outro brasileiro que possua o mesmo número. Portanto o CPF é uma excelente chave.
- Identifique os relacionamentos entre as entidades.
- Por fim, determine a cardinalidade.

Agora pense no caso da Clínica Bem-Estar visto na unidade 2, exercício das atividades de auto-avaliação. Vamos identificar pelos menos 3 entidades:

- Paciente;
- Médico;
- Agenda_Consultas.

Neste caso o relacionamento seria:

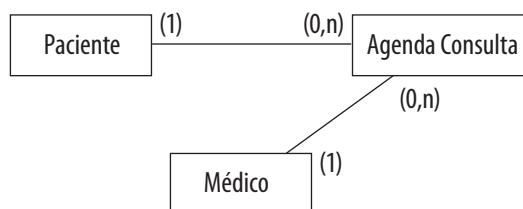



Figura 3.16 - Exemplo de Cardinalidade Clínica Bem-Estar

Onde um paciente pode ter nenhuma, uma, ou várias consultas agendadas. Mas, para cada consulta só vai haver um paciente. Para cada consulta, vai haver relacionado um médico, mas um médico pode ter várias consultas agendadas.

Agora para praticar os conhecimentos conquistados nesta unidade, realize as atividades propostas a seguir

Atividades de auto-avaliação

Leia com atenção os enunciados e, após, realize as questões propostas:

1) Relacione os conceitos abaixo, observando que uma mesma opção pode se repetir:

A. Diagrama de Fluxo de dados B. Dicionário de Dados C. Descrição dos Processos D. Entidade Externa E. Fluxo de Dados F. Depósito de Dados G. Processo	a. ( ) Notação utilizada para descrever o conteúdo e significado de fluxos, entidades externas, processos e depósitos de dados. b. ( ) Notação que permite representar um processo em um DFD c. ( ) Utilizado para representar o cliente em um DFD d. ( ) Utilizado para representar um repositório de dados em um DFD e. ( ) Permite descrever os fluxos de informação e as transformações sofridas pelos dados durante a execução do processo f. ( ) Utilizado para representar os dados que serão inseridos no processo em um DFD g. ( ) Utilizado para representar em um DFD uma empresa que participa de alguma maneira no processo, por exemplo, uma instituição bancária em um processo de contas a receber.

2) Defina a cardinalidade e as entidades existentes para as situações propostas abaixo:

a) Um professor leciona várias disciplinas em sua universidade.

b) A universidade emprega vários funcionários.

c) Os funcionários são lotados em um departamento.

d) Um aluno pode estar matriculado em nenhuma ou várias disciplinas e uma disciplina pode ter vários alunos nela matriculados.

Exemplo de resolução: um equipamento de audiovisual é alocado a um professor

Síntese

Nesta unidade você estudou os principais conceitos adotados nesta metodologia. Durante o estudo o uso das ferramentas, como o DFD, o dicionário de dados, a descrição do processo e o diagrama ER foram abordados, evidenciando sua importância na concepção do modelo previsto para a solução do problema do cliente. O uso do DFD torna clara a equipe de projeto e, para o usuário, a forma como as informações irão transitar nos futuros processos assim como a transformação sofrida durante sua evolução. Foi possível também abordar a importância da normalização evitando redundâncias na futura base de dados do sistema.

A análise estruturada foi um movimento inicial do uso de uma metodologia de projeto que permitisse documentar o futuro projeto de forma clara e consistente.

A evolução, no entanto, era inevitável. Na próxima unidade abordaremos a análise orientada a objetos, que traz inovações à análise tradicional aproximando-a ainda mais do cliente.

Saiba mais

Para conhecer um pouco mais sobre a análise estruturada, você deve dar uma olhadinha nos seguintes livros:

- YOURDON, Edward. **Análise estruturada moderna.** Rio de Janeiro: Campus, 1992.
- DEMARCO, Tom. **Análise Estruturada e Especificação de Sistema.** Rio de Janeiro: Campus, 1989.

No EVA, por meio da ferramente Midiateca, você vai encontrar exemplos práticos utilizando a metodologia estruturada.

Mas, para implementar os diagramas da análise estruturada, será preciso ferramentas. Conheça algumas visitando os sites:

- http://www.sparxsystems.com.au
- http://www.otw*software*.com/english/index.shtml
- http://www.rational.com/products/rose/index.jsp
- http://www.ilogix.com
- http://argouml.tigris.org
- http://www.gentleware.com/products/download.php3

UNIDADE 4

4

Visão geral da UML

Objetivos de aprendizagem

- Compreender as diferenças fundamentais existentes entre a análise estruturada e a análise orientada a objetos.
- Perceber as diferentes visões da UML e os diagramas oferecidos para viabilizar seu entendimento.
- Conhecer a história e o surgimento da linguagem de modelagem UML e suas diferentes possibilidades de aplicação.

Seções de estudo

Seção 1 O Paradigma da orientação a objetoss

Seção 2 Qual a origem das linguagens de modelagem?

Seção 3 Visões de um sistema

Seção 4 Diagramas da UML

Seção 5 Ferramentas

Para início de estudo

Na quarta unidade você vai iniciar seu estudo acerca da análise orientada a objetos.

Para compreender este conceito é necessário que você perceba suas diferenças em relação à análise estruturada e suas diferentes visões relacionadas ao processo de desenvolvimento.

Você pode usar a UML para modelar diferentes fases de um sistema, desde a análise do problema até a geração do código. A linguagem pode ser aplicada em qualquer tipo de sistema de desenvolvimento de *software* ou mesmo em sistemas mecânicos de engenharia ou ainda na organização de processos de uma organização.

A UML é hoje uma das abordagens mais utilizadas no mundo, esse sucesso se deve, em parte, por sua padronização, facilidade de reutilização, flexibilidade e possibilidade de abstração de dados.

A partir desta unidade você vai submergir, aos poucos, no mundo orientado a objetos, usando conceitos e ferramentas.

SEÇÃO 1 - O paradigma da orientação a objetos

Durante a unidade 3 você percebeu que a principal ênfase sempre é dada aos procedimentos. Os procedimentos são implementados em blocos e a comunicação entre eles se dá pela passagem de dados. Na orientação a objetos, os dados e procedimentos fazem parte de um só elemento básico; isso significa que se encontram encapsulados em um só elemento.

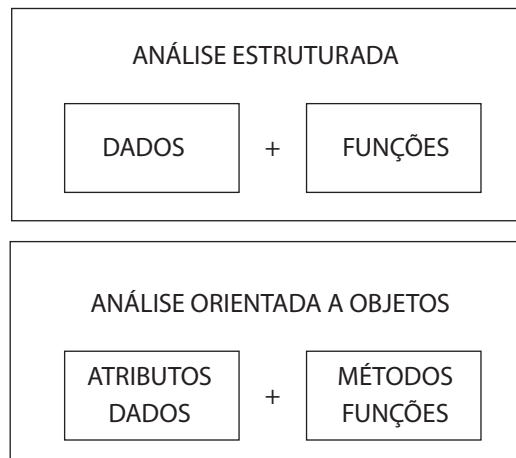



Figura 4.1 - Visão comparativa da Análise
Fonte: Pacheco (1994).

Pode-se dizer que, na visão estruturada, a perspectiva adotada é a de um algoritmo, em que o bloco principal de construção do *software* é o procedimento ou a função. A visão do desenvolvedor será a de decompor algoritmos maiores em menores.

Assim, você pode dizer que a análise estruturada desenvolve uma visão de desenvolvimento baseada em um modelo entrada-processamento-saída. Com o passar do tempo, a manutenção pode tornar-se difícil, pois o sistema é totalmente construído a partir do foco do algoritmo.

Quando se adota a visão orientada a objetos, o bloco de construção do *software* passa a ser o objeto ou a classe. Mas, você sabe o que é um objeto?

O objeto é uma abstração de conjunto de coisas do mundo real; pode ser uma máquina, uma organização,um carro, uma passagem de ônibus.

A figura Objetos apresenta "coisas" do mundo real, e, portanto são objetos sob o ponto de vista da orientação a objetos.

Figura 4.2. Objetos

E uma classe? A classe pode ser vista como a descrição de um tipo de objeto, com propriedades semelhantes (atributos), o mesmo comportamento (operações), os mesmos relacionamentos com outros objetos e a mesma semântica. Por exemplo: a classe aluno de uma escola, com um conjunto de alunos que apresentam as mesmas informações. Todos os objetos são instâncias de classes. A classe, por sua vez, deve descrever as propriedades e comportamentos daquele objeto.

Uma classe descreve um grupo de objetos.

Observe que os objetos apresentados na figura 4.2 podem ser agrupados por apresentarem atributos, características ou comportamentos semelhantes. Isto permite que sejam criadas as classes Animais, Edificações e Transportes.

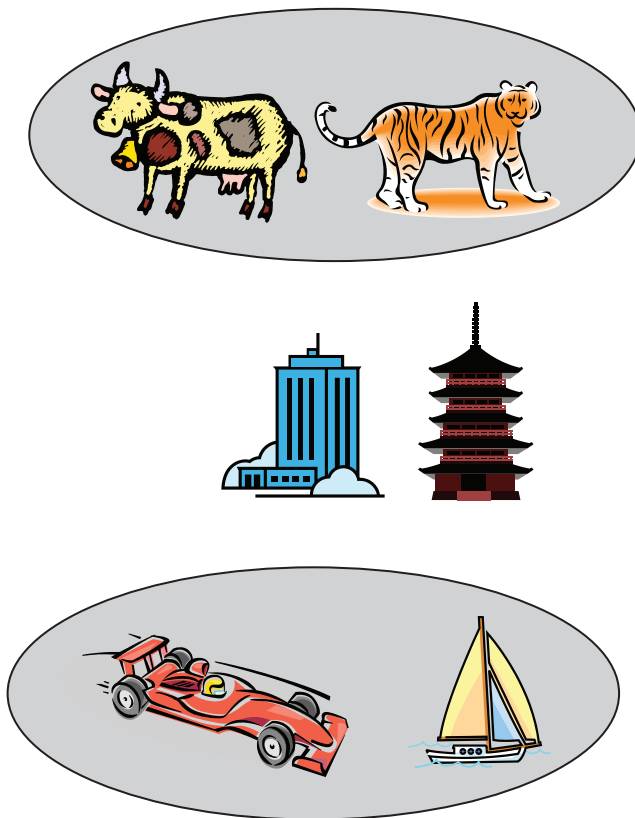



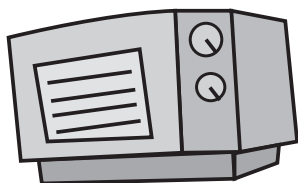
Figura 4.3. Agrupamento de Objetos em Classes

Observe que é possível perceber atributos comuns assim como características e comportamentos e, por este motivo, é possível agrupar estes objetos.

Veja na classe animais pode se indicar alguns atributos como espécie, data nascimento, nome que são comuns a qualquer animal, assim como podemos indicar comportamentos comuns como comer, correr, dormir. Esta proximidade é o primeiro passo na identificação de uma classe.

A orientação a objetos pressupõe que o mundo é composto por objetos, sendo um objeto uma entidade que combina estrutura de dados e comportamento funcional. No paradigma orientado a objetos os sistemas são estruturados a partir dos objetos que existem no domínio do problema, isto é, os sistemas são modelados como um número de objetos que interagem. Mas, o que você entende pela expressão "orientado a objetos"?

Se você teve dificuldade em conceituar não se preocupe, mesmo *experts* nessa área passaram anos engalfinhando-se, tentando

esclarecer seu significado. PAGE-JONES (2001), tentando explicar, lista seus principais conceitos, que juntos explicam o método:

- **Encapsulamento** – é o agrupamento de idéias afins em uma unidade. Permite ao programador esconder os detalhes da representação dos dados por trás de um conjunto de operações (como a interface). Reflita sobre o seguinte: você sabe como funciona internamente o seu microondas? Mas, você sabe como ligá-lo, programá-lo e desligá-lo. Você interage com o microondas por meio de sua interface sem se preocupar com os detalhes da implementação: este é um exemplo de encapsulamento.
- **Ocultação de informações e implementações** – é exatamente o uso do encapsulamento que restringe a visibilidade de detalhes e informações que ficam internas à estrutura do encapsulamento.
- **Mensagens** - solicitação de um objeto para que outro objeto efetue uma de suas operações. Os objetos mandam mensagens entre si. As mensagens resultam na chamada de métodos que executam as ações necessárias.
- **Classes** – as classes são conjuntos de objetos com características semelhantes.
- **Herança** - o mecanismo de herança permite que uma classe seja criada a partir de outra classe (superclasse), e a nova classe (subclasse) herda todas as suas características.
- **Poliformismo** - é a propriedade segundo a qual uma operação pode se comportar de modos diversos em classes diferentes.

Sistemas orientados a objetos são flexíveis a mudanças, possuem estruturas bem conhecidas e oferecem a oportunidade de criar e implementar componentes totalmente reutilizáveis.

Quer conhecer mais?

Para saber um pouco mais sobre orientação a objetos e suas características, leia o capítulo 1 do livro Fundamentos do Desenho Orientado a Objetos de Pages-Jones (Editora Makron Books, 2001).

SEÇÃO 2 - Qual a origem das linguagens de modelagem?

As linguagens de modelagem orientada a objetos começaram a surgir na década de 70. Nessa década, também apareceram no mercado computadores mais modernos e acessíveis, iniciando uma grande expansão computacional.

No início da década de 90, surgem as primeiras metodologias orientadas a objeto. Muitos foram os métodos oferecidos como solução para especificação de projetos orientados a objetos, mas alguns se destacaram tornando-se referência por suas características:

- **O método Booch (1993)** – considerado expressivo nas fases de projeto e construção, apresenta o projeto como um conjunto de visões, cada visão pode ser descrita por modelos e diagramas. A simbologia do modelo é bastante complexa.
- **O método OOSE (*Object Oriented* *Software* *Engineering*) (1993) de Jacobson** – excelente no controle da captura de requisitos, análise e projeto de alto nível. Utiliza-se de *use-cases* para definir os requisitos iniciais do sistema, vistos por um ator externo.
- **O método OMT (*Object Modeling Technique*) (1995) de Rumbaugh** – bastante usado para análise e sistemas de informações com uso intensivo de dados. Possui um forte foco para o teste de modelos, baseado nas especificações da análise de requisitos do sistema.

Além dos métodos citados, existiam muitos outros; cada método utilizava-se de notações diferentes. Com o passar do tempo a percepção de que os métodos poderiam ser complementares a partir de suas melhores características, fez com que os três autores se unissem na especificação de um novo método, proporcionando estabilidade e uma linguagem clara e madura que auxiliasse com problemas que, provavelmente, nenhum dos três métodos poderia resolver.

Qual a origem da Linguagem de Modelagem Unificada – UML?

No ano de 1994 nas dependências da *Rational Software*, iniciou-se a junção do método Booch e OMT. Em 1995, Jacobson se juntou à equipe e decidiram a incorporação do método OOSE. A primeira versão (0.9) da UML foi lançada ao mercado em 1996. A partir desse lançamento, profissionais da área contribuíram com críticas e sugestões ao modelo. Empresas, como a Hewlett-Packard, I-Logix , DEC, IBM, Microsoft, Oracle Texas, entre outras, investiram maciçamente no aperfeiçoamento do método.

Os pesquisadores Booch, Jacobson e Rumbaugh foram os idealizadores do projeto, mas o produto final em sua versão 1.3 foi resultado de um trabalho de equipe por meio da participação de diversos colaboradores, contribuindo com suas experiências e seu ponto de vista.

Em 1997, a UML foi aprovada como padrão pela OMG – *Object Management Group*.

A UML é uma linguagem completamente visual, por meio de elementos gráficos ela permite a representação de conceitos da orientação a objetos utilizados na estrutura de elaboração de um projeto de *software*.

Você pode representar o sistema por diferentes perspectivas dependendo do diagrama que você irá utilizar. Cada um de seus diagramas possui uma forma pré-determinada de desenhar o elemento, uma semântica que define o significado deste elemento e onde o mesmo pode ser utilizado.

Não existe dependência de linguagem no uso da UML ou processos de desenvolvimento. Ela pode servir para qualquer linguagem que o desenvolvedor venha a utilizar.

Além de ser utilizada como uma linguagem de especificação para construir modelos, a UML oferece a possibilidade de conectar seus modelos a diversas linguagens como JAVA, C++, Visual Basic e, inclusive, bancos de dados. Em outras palavras, você pode gerar código a partir de um modelo UML em uma linguagem de alto nível como JAVA, por exemplo.

SEÇÃO 3 - Visões de um sistema

Na UML todas as abstrações de um sistema são organizadas em modelos, onde, cada um desses modelos representa uma visão do sistema.

Se você olhar para o desenvolvimento como um todo, perceberá etapas bem definidas; cada etapa pode ser representada por meio de diagramas e modelos de elementos, de forma a proporcionar ao projetista uma visão eficiente e completa daquela etapa. Então, você pode assumir que as Visões mostram diferentes aspectos do sistema que está sendo modelado.

A visão vai apresentar um aspecto particular do sistema.

Segundo BEZERRA (2002), a UML pode ser apresentada a partir de cinco visões:

- **Visão de casos de uso** - descreve a funcionalidade do sistema desempenhada pelos usuários. A visão *use case* é central, pois seu conteúdo é fundamental para a composição das demais visões do sistema.
- **Visão de Projeto** – são enfatizadas as características do sistema que dão suporte estrutural e comportamental.
- **Visão de Implementação** – abrange o gerenciamento de versões do sistema construídas através do agrupamento de módulos e subsistemas.
- **Visão de Implantação** – corresponde à distribuição física do sistema em seus subsistemas e a conexão entre essas partes.
- **Visão de Processo** – esta visão enfatiza as características da concorrência, sincronização e desempenho do sistema.

O uso das visões durante o projeto depende do tipo de projeto a ser construído. A visão do processo, por exemplo, é irrelevante se o sistema for construído sobre apenas um processo. Se o sistema é composto por apenas um módulo, é redundante a utilização da visão de implementação.

A empresa decide pela escolha das visões que serão contempladas e, conseqüentemente, os diagramas, que serão desenvolvidos no projeto a partir do tipo de domínio do projeto, modelo de desenvolvimento, riscos, restrições e características do processo.

SEÇÃO 4 - Diagramas da UML

Um diagrama procura representar graficamente a projeção de um sistema. Alguns elementos aparecem em mais de um diagrama, alguns em todos; outros em nenhum.

A UML possui nove diagramas (FURLAN, 1998):

a) Diagrama de Casos de Uso - Os casos de uso descrevem a funcionalidade do sistema percebida por atores externos. O ator (um usuário, equipamento, organização) interage com o sistema.

b) Diagrama de Classes - representa a estrutura estática do sistema, as classes, os relacionamentos entre suas instâncias (objetos), restrições e hierarquias.

c) Diagramas de Interação – é formado por:

- Diagramas de Seqüência: mostram a colaboração dinâmica entre um número de objetos. O objetivo principal é mostrar a seqüência de mensagens enviadas entre objetos.
- Diagramas de Comunicação: têm o mesmo propósito dos diagramas de seqüência. São desenhados como diagramas de objetos, onde são mostradas as mensagens trocadas entre os objetos.

d) Diagramas de Estados - mostram as seqüências de estados do objeto a partir dos estímulos recebidos, suas respostas e ações. O Diagrama de atividades é uma variação dos Diagramas de Estado, em que a maioria dos estados é estado de ação, e a maioria das transições é ativada por conclusões das ações.

e) Diagramas de Implementação - é composto por dois diagramas:

- Diagrama de Componentes: são mostradas as dependências entre componentes de *software* (código fonte, código binário e componentes executáveis).
- Diagrama de Implantação: mostra elementos de configuração do processamento em tempo de execução, os componentes de *software*, processos e dispositivos físicos.

Os diagramas da UML podem ser situados a partir de suas cinco visões (BOOCH, 2000):

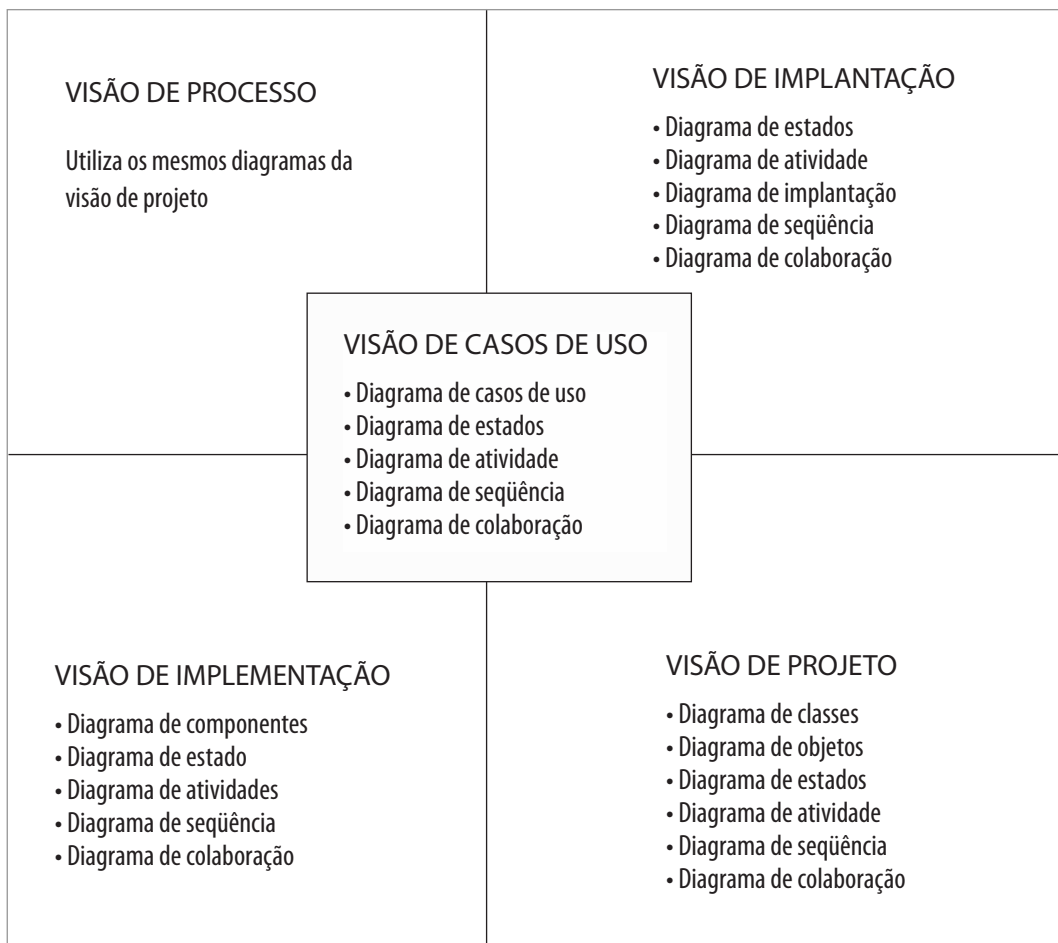



Figura 4.2 - As 5 visões dos diagramas da UML

Os aspectos estáticos do sistema são capturados pelos diagramas de classes, de objetos, de componentes e de casos de uso; os aspectos dinâmicos, pelos diagramas de estado, de atividade, de seqüência e de colaboração. O modelo funcional é suportado pelos diagramas de componente e execução.

SEÇÃO 5 - Ferramentas

A escolha de alguma destas ferramentas é fundamental para que você continue seus estudos.

Para modelar um sistema utilizando a notação UML, é fundamental que você utilize uma **ferramenta** que automatize o método.

Como a UML utiliza-se de uma notação gráfica, uma boa ferramenta agiliza o processo de construção e recuperação da informação.

As ferramentas disponíveis subdividem-se na categoria *software* livre e demos (são ferramentas pagas que oferecem 30 dias para avaliação).

A seguir estão listadas algumas delas:

a) *Software* Livre:

- **Orquídea** - http://www.umlnapratica.com.br/orquidea/orquidea.php

Orquídea é uma ferramenta *CASE* que possui as seguintes funcionalidades: construção de diagrama de classes e de diagramas de seqüência na notação UML, geração e leitura de código C++; geração de documentação *web* em HTML ou HTM.

- **EclipseUML** - http://www.eclipse.org

Eclipse é uma plataforma aberta para integração de ferramentas criada por uma comunidade aberta de provedores de ferramentas. A IDE Eclipse foi criada sobre o paradigma *Open Source* baseada na *Common Public License*.

- **Omondo EclipseUML** - http://www.omondo.com

Omondo EclipseUML é um *plugin* para o Eclipse que auxilia na construção de diagramas UML. Com este *plugin* você pode criar diagramas de classe, seqüência, estados, *use cases*, atividades, etc. Alterações no diagrama automaticamente se refletem no código-fonte e vice-versa.

b) Versões Demo:

- ***Rational Rose*** - http://www-306.ibm.com/*software*/rational/

Ferramenta de modelagem que suporta todos os diagramas previstos na linguagem de modelos UML. Seu custo é extremamente alto, sua grande vantagem é que ela pertence à *Rational*, originalmente criadora da linguagem UML.

- **EA - *Enterprise Architect*** - http://www.sparxsystems.com.au/ea_downloads.htm

Ferramenta da Sparxsystems da Austrália, suporta todos os diagramas previstos pela UML e toda notação da OMG. Gera códigos e engenharia reversa para manutenção dos modelos. Apresenta uma interface mais amigável e simples de se usar.

- **Poseidon** - http://www.gentleware.com/products/descriptions/ce .php4

Baseada no mesmo *engine* do ArgoUML, mas com melhorias, possui uma *Comunity Edition* gratuita. Não permite funcionalidades de impressão de diagramas, mas possibilita a exportação das imagens dos diagramas.

A partir da próxima unidade, você vai iniciar a confecção de diagramas, sendo necessário o uso de uma ferramenta. Todas as **ferramentas** apresentam, nas respectivas páginas de download, tutoriais e manuais de instalação e utilização.

Na midiateca você também vai encontrar documentação sobre algumas das ferramentas.

Quer conhecer mais?

Se você quer aprofundar seu conhecimento sobre as diferentes visões da UML, leia o Capítulo II – Introdução a UML do livro UML Guia do Usuário, escrito pelos autores BOOCH, RUMBAUGH, JACOBSON, I. (Ed. Campus/2000).

Agora, para praticar os conhecimentos conquistados nesta unidade, realize as atividades propostas a seguir.

Atividades de auto-avaliação

Leia com atenção os enunciados e, após, realize as questões propostas:

1) É correto afirmar que (pode haver mais de uma afirmativa correta):

a) ( ) A análise estruturada possui como fundamento a visão da funcionalidade. Assim, a visão que se impõe ao modelo é o processamento dos dados.

b) ( ) Na análise orientada a objetos, dados e métodos são vistos como uma entidade única, preocupando-se com propriedades e comportamentos do objeto.

c) ( ) A análise estruturada possui como fundamento a visão do comportamento do processo. Assim, a visão que se impõe ao modelo é o processamento dos dados.

d) ( ) A análise orientada a objetos é construída a partir de objetos. As classes são instâncias do objeto.

2) Identifique os elementos a seguir **[C]**lasse, **[O]**bjetos:

a) ( ) Caixa

b) ( ) Imposto Pago

c) ( ) João da Silva

d) ( ) Valor Venda

e) ( ) Cliente

3) Observe os conceitos abaixo e realize as devidas inserções:

Encapsulamento Herança Poliformismo Mensagens

a) __________________pode explicar situações nas quais pode haver várias formas de fazer uma determinada "coisa".

b) __________________permite que detalhes internos sejam "escondidos".

c)____________________especifica informações a serem passadas para a operação que deve ser executada por um objeto receptor.

d) Você define uma classe chamada Conta (para um sistema bancário) com características e comportamento genérico. Posteriormente, você define duas classes, chamadas Poupança e Conta_Corrente; cada uma delas possui propriedades específicas que a outra não possui, mas agregam a elas o comportamento genérico da classe Conta. Isso é chamado de ________________.

4) As visões da UML permitem o uso de diferentes diagramas, adaptando o uso do diagrama às necessidades do projeto. Relacione conceitos e diagramas utilizados ao tipo de visão associado:

A) Visão de casos de uso

B) Visão de Projeto

C) Visão de Implementação

D) Visão de Implantação

E) Visão de Processo

a) ( )	Descreve a funcionalidade do sistema desempenhada pelos usuários
b) ( )	A visão é representada por meio dos diagramas de classe, de objetos, de estados, de atividade, de seqüência e de colaboração.
c) ( )	Corresponde à distribuição física do sistema em seus subsistemas e a conexão entre essas partes.
d) ( )	Abrange o gerenciamento de versões do sistema, construídas por meio do agrupamento de módulos e subsistemas.
e) ( )	A visão é representada por meio dos diagramas de casos de uso, de estados, de atividade, de seqüência e de colaboração.
f) ( )	Enfatiza as características da concorrência, sincronização e desempenho do sistema.
g) ( )	São enfatizadas as características do sistema que dão suporte estrutural e comportamental.

Síntese

Na unidade quatro você estudou as diferenças existentes entre a análise estruturada e a análise orientada a objetos. Você percebeu que a UML surgiu a partir da evolução de outros métodos orientados a objetos e que, de certa maneira, se complementaram dentro de uma linguagem de modelagem padronizada.

Ao estudar a unidade apresentaram-se as diferentes visões que, de diferentes formas, apóiam os projetistas na melhor visualização de todas as etapas do ciclo de desenvolvimento do *software*. Além da introdução conceitual, da análise orientada a objetos e à UML, também foram apresentadas algumas ferramentas que suportam o método.

Na próxima unidade você vai iniciar o processo de modelagem por meio da concepção dos casos de uso de um sistema. Um caso de uso procura especificar uma seqüência de interações entre um sistema e os agentes externos que fazem uso do sistema.

Saiba mais

Para aprofundar as questões abordadas nesta unidade, você poderá pesquisar em:

- GILLEANES, T. A. Guedes. **UML**: uma abordagem prática. Novatec, 2004.

Você também poderá acessar site de UML http://www.uml.org

Nesse site são apresentados padrões, organizações, certificações e estudos de caso.

UNIDADE 5

5

Modelagem de casos de uso

Objetivos de aprendizagem

- Compreender a importância do uso de casos de uso para identificação clara dos objetivos do usuário.
- Entender o significado dos diferentes elementos existentes em um caso de uso.
- Reconhecer meios para identificar atores e casos de uso.
- Compreender o mecanismo de documentação e sua importância na descrição dos casos de uso.

Seções de estudo

Para início de estudo

Nesta unidade você vai iniciar o estudo sobre os diagramas oferecidos na UML e suas representações utilizadas na modelagem orientada a objetos.

Este estudo inicia com os casos de uso que representam o eixo central do modelo, pois descrevem a seqüência de interações realizadas pelo sistema que visam prover algum valor mensurável ao usuário do sistema. O caso de uso permite mapear o escopo do sistema facilitando a comunicação com o usuário do sistema e facilitando a gerência do projeto.

Mas, para descrever o caso de uso não é suficiente concebermos apenas o diagrama de caso de uso, é preciso entender todos os seus elementos e sua documentação.

Nesta unidade você vai iniciar seu processo de aprendizagem identificando e construindo os casos de uso necessários para descrever esta visão do projeto.

SEÇÃO 1 - O que são casos de uso?

Um sistema sempre interage com usuários, outros sistemas ou equipamentos. Todos eles esperam pelos resultados desta interação que normalmente são previsíveis, ou pelo menos deveriam ser.

Um caso de uso procura documentar as ações necessárias, comportamentos e seqüências para que o resultado esperado pelo usuário ocorra. Então você pode dizer que modelo de casos de uso modela os requisitos funcionais do sistema.

Antes de iniciar a definição dos casos de uso você deve estar consciente dos requisitos funcionais e não-funcionais necessários ao sistema (Unidade 02).

Como o caso de uso não especifica detalhes de implementação, você vai, na verdade, pensar em como o sistema será utilizado.

Agora imagine a seguinte situação: você vai realizar o projeto usando UML, para uma pequena videolocadora. Neste pequeno projeto os requisitos funcionais listados são:

- o gerente deseja cadastrar seu acervo de fitas e dvd's com informações gerais sobre cada filme, como: nome do filme; duração; diretor; principais atores; gênero; idiomas disponíveis.
- É fundamental que o atendente possa realizar o cadastro de clientes possibilitando o cadastro de endereço, telefone do cliente e 3 possibilidades para digitação de nomes de pessoas autorizadas para retirar fitas/DVDs.
- É necessário que o sistema permita o controle de entrega e recebimento das fitas/DVD.
- O gerente deseja um relatório estatístico de fitas/DVDs mais locados
- Deve ser possível um relatório de consulta de fitas/DVDs em atraso.
- Relatório que apresente os 50 maiores clientes da locadora.

Agora tente identificar os casos de uso, então quantos você encontrou?

Você pode listar os seguintes casos de uso:

- Gerenciar cliente – este caso de uso será responsável pelo cadastro, consulta, exclusões e alterações de dados cadastrais do cliente.
- Gerenciar filmes – este caso de uso será responsável pelo cadastro, consulta, exclusões e alterações de dados das fitas e dvd's da videolocadora.
- Gerenciar locações – o caso de uso deve permitir o controle de entregas e recebimentos das fitas/DVDs e cálculo de multas quando necessário.

- Gerenciar relatórios – este caso de uso deve possibilitar a impressão dos relatórios estatísticos de locação, maiores clientes e controles de atraso de entrega.

Se você representar graficamente os casos de uso você terá os mesmos representados por um círculo, veja a figura 5.1.

Retome o exemplo da videolocadora. A figura 5.1 mostra três casos de uso possíveis para este projeto.

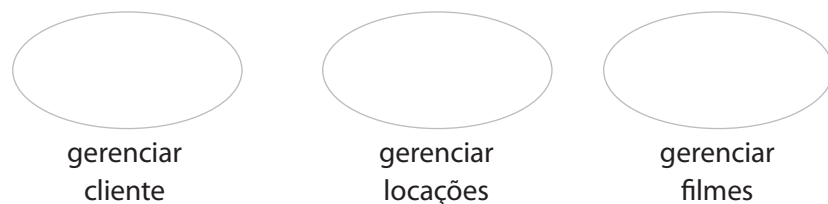



Figura 5.1 - Casos de uso de uma videolocadora

O modelo de casos de uso é composto por casos de uso, atores e relacionamentos.

O caso de uso descreve um conjunto de seqüências, cada um representando a interação entre atores com o próprio sistema. Esses comportamentos são funções em alto nível que você vai visualizar, especificar, construir e documentar tentando mostrar de forma clara o comportamento pretendido do sistema durante a análise de requisitos.

Quando você nomear o caso de uso, este nome deve ser único dentro do projeto, diferenciando-o dos demais casos de uso. Você pode usar qualquer caractere textual no nome do caso de uso, mas evite o uso de dois-pontos. Procure utilizar expressões verbais ativas.

O que é o diagrama de casos de uso?

Se um caso de uso é sempre iniciado a partir do momento que um ator envia sua mensagem (estímulo), o diagrama de casos

de uso é criado para visualizar os relacionamentos entre atores e casos de uso.

O diagrama de casos de uso pode ser utilizado para representar um caso de uso e seus relacionamentos, todos os casos de uso para um ator ou ainda todos os casos de uso a serem implementados em um ciclo de desenvolvimento.

A notação do diagrama faz uso de um boneco para identificar o ator, abaixo do boneco é inserido o nome do ator cliente. A elipse identifica o caso de uso, o nome do caso de uso pode ser escrito no interior ou externo a elipse (caso de uso: gerenciar cliente). A linha reta indica o relacionamento de comunicação entre o ator e o caso de uso.

Figura 5.2 - Caso de uso Gerenciar Cliente

SEÇÃO 2 - Como identificar os atores?

Um ator representa um conjunto coerente de papéis que os usuários de casos de uso desempenham quando interagem com esses casos de uso. Um ator pode representar um papel que um ser humano, um dispositivo de *hardware* ou até outro sistema que desempenha alguma interação com o sistema (BOOCH, 2000).

O ator troca informações com o sistema, ele não faz parte do sistema apenas interage com ele. Um ator pode participar de vários casos de uso.

Você pode ter em seu sistema diferentes tipos de atores:

- pessoas (professor, aluno, secretária, coordenador)
- organizações (empresa fornecedora, bancos, receita federal)

- outros sistemas (módulos que venham a interagir com seu sistema, como contas a pagar, crediário, estoque).
- equipamentos (coletor de dados, leitora de código de barras, balanças).

Quando você escolhe o nome do ator lembre-se de que ele representa um papel no sistema, nunca utilize nomes pessoais. Imagine: um indivíduo pode representar o papel de funcionário em alguns momentos e em outros pode representar o papel de gerente.

Neste caso nomes que representam o papel poderiam ser Gerente, Professor, Vendedor, Fornecedor.

Mas, como identificar os atores do sistema?

Primeiro você deve identificar quais as fontes de informação que serão processadas e quais os destinos das informações geradas.

Sempre avalie quais os departamentos da empresa que serão afetados e que potencialmente podem ter atores que interagem com o sistema.

Bezerra (2002) oferece algumas dicas na forma de perguntas que apóiam a descoberta dos atores do sistema:

- Que órgãos, empresas ou pessoas utilizarão o sistema?
- Que outros sistemas irão se comunicar com o sistema a ser construído?
- Alguém deve ser informado de alguma ocorrência no sistema?
- Quem está interessado em um certo requisito funcional do sistema?

Figura 5.3 - Exemplos de Atores

Você consegue listar os atores da videolocadora? Pense um pouco sobre o assunto.

Os possíveis atores são:

- O atendente
- O gerente
- O cliente

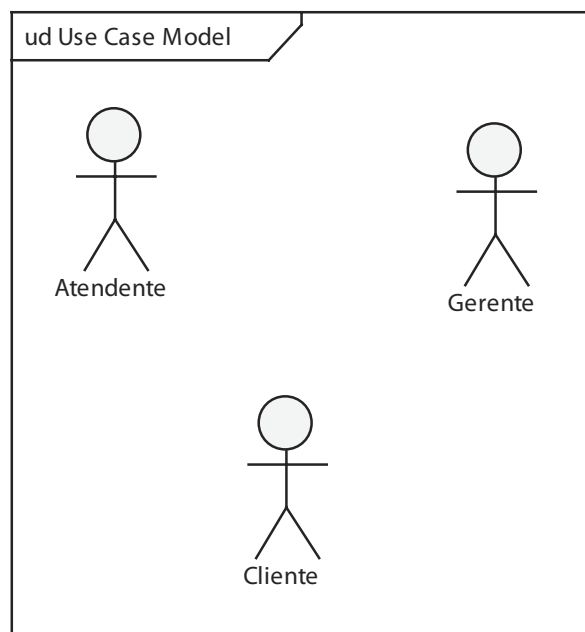



Figura 5.4. Diagrama de Atores da Vídeo Locadora

Só definir os atores não é o suficiente, você deve descrever características e responsabilidades destes atores. Características importantes podem ser cargos, funções, grau de escolaridade, permissões de acesso, freqüência de uso, conhecimento em informática, conhecimento no processo do negócio.

Os atores podem ser divididos em primários e secundários, o ator primário inicia a seqüência de ações de um caso de uso. Já os atores secundários supervisionam, operam, mantêm ou auxiliam na utilização do sistema.

Tabela 5.1 - Exemplo de descrição de atores

Nro.	Ator	Descrição
1	Cliente	• Definição – indivíduo que realiza locações de fita na videolocadora. • Freqüência de uso – diário,semanal • Conhecimento em informática – relativo, alguns clientes possuem outros não • Conhecimento no processo – sim, à grande maioria possui uma noção clara do funcionamento do processo de locação de fitas • Grau de escolaridade – desde fundamental a pós-graduação • Permissões de acesso – deve ser disponibilizado ao cliente a consulta ao acervo.
2	Gerente	• Definição – funcionário da videolocadora responsável por operações de abertura, fechamento, controle de funcionário, controle de compras e pagamentos da videolocadora. • Freqüência de uso – diário • Conhecimento em informática – aplicativos *Word, Browsers, Windows* XP • Conhecimento no processo – domina todo o processo do negócio • Grau de escolaridade – graduação • Permissões de acesso – terá acesso a todas as funcionalidades do sistema

A descrição detalhada do ator ajuda na definição de perfis que influenciam no projeto da interface.

SEÇÃO 3 - Relacionamentos entre casos de uso e atores

Para que um caso de uso seja executado é necessário a existência de atores que interajam com o caso de uso. Esta interação ocorre por meio dos relacionamentos de comunicação, inclusão, extenção e generalização.

O que é relacionamento de comunicação?

Quando se usa um relacionamento de comunicação você vai estar representando quais atores estão associados a um caso de uso. Isto significa que o ator vai interagir trocando informações com o caso de uso. É o relacionamento mais comum onde ocorre troca de informações.

Figura 5.5 - Exemplo de relacionamento de comunicação

O ator cliente oferece informações ao caso de uso como nome e título do filme que deseja, já o ator atendente oferece ao caso informações como o código.

Agora observe o diagrama geral de casos de uso a partir dos requisitos funcionais apontados na seção 1.

O diagrama geral apresenta de forma genérica os casos de uso solicitados sendo que todo o relacionamento existente é de comunicação.

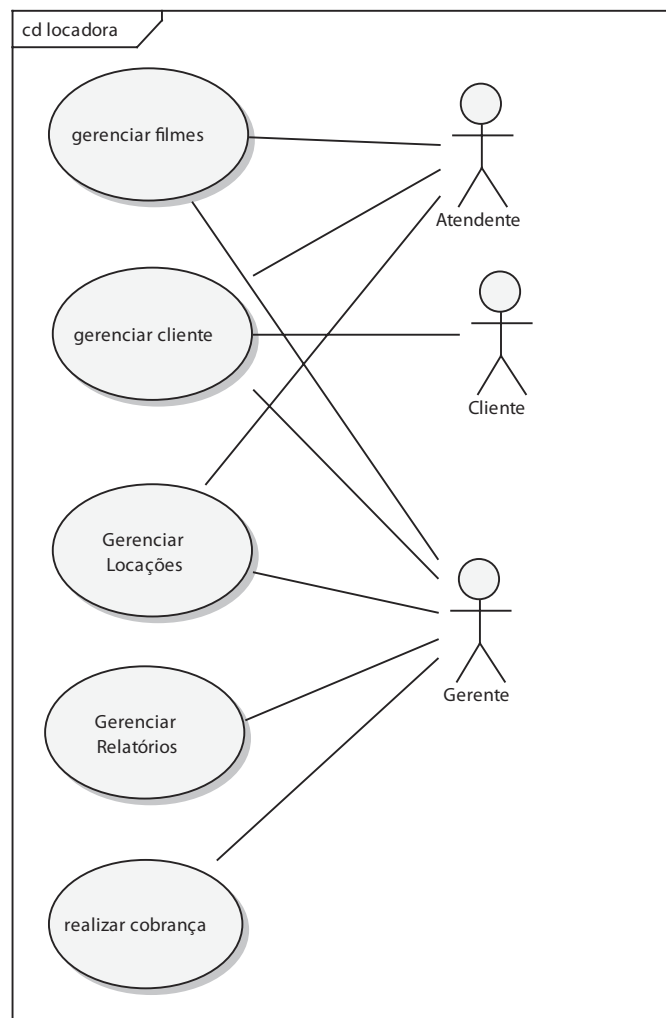



Figura 5.6. Diagrama de Casos de Uso da Vídeo Locadora

Os casos de uso possuem na maioria das vezes um fluxo principal e vários fluxos alternativos. Os casos de uso secundários são muitas vezes inseridos no caso de uso por meio do relacionamento de extensão.

O relacionamento de extensão deve ser usado para representar:

- um comportamento opcional;
- um comportamento que só ocorre sob certas condições (alarmes por exemplo);

- em fluxos alternativos dependentes da escolha de um ator.

Para entender melhor considere a seguinte situação:

Quando o ator decide executar o caso de uso extensor, ele executa e após sua execução retorna ao caso de uso estendido.

Você pode comparar um caso de uso extensor a uma função que você desenvolve em um algoritmo de programação.

Veja a seguinte situação:

Em nosso estudo de caso da Vídeo Locadora, o contratante do projeto deseja a inserção no cadastro do cliente das preferências do cliente, por exemplo, as preferências do cliente a respeito do tipo de filme (comédia, romance,ficção, etc), diretores, atores.

Para você inserir esta funcionalidade na modelagem dos casos de uso segue o seguinte raciocínio:

- O caso de uso gerenciar cliente vai descrever a seqüência de interações para inserir os dados cadastrais do cliente.
- Normalmente o cliente informa apenas as informações cadastrais e salva seus dados, este é então o fluxo normal de interações.
- MAS caso o cliente deseje, ele pode ainda inserir informações sobre as suas preferências relacionadas aos filmes.
- Esta possibilidade de inserir preferências é opcional e depende da solicitação do usuário, então pode ser definido como um caso de uso estendido.

Veja como este relacionamento é representado :

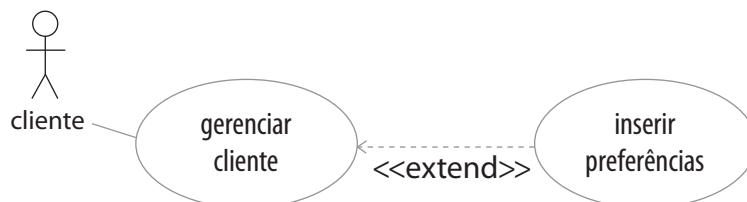



Figura 5.7 - Relacionamento de Extensão

O diagrama geral de casos de uso atualizado seria então:

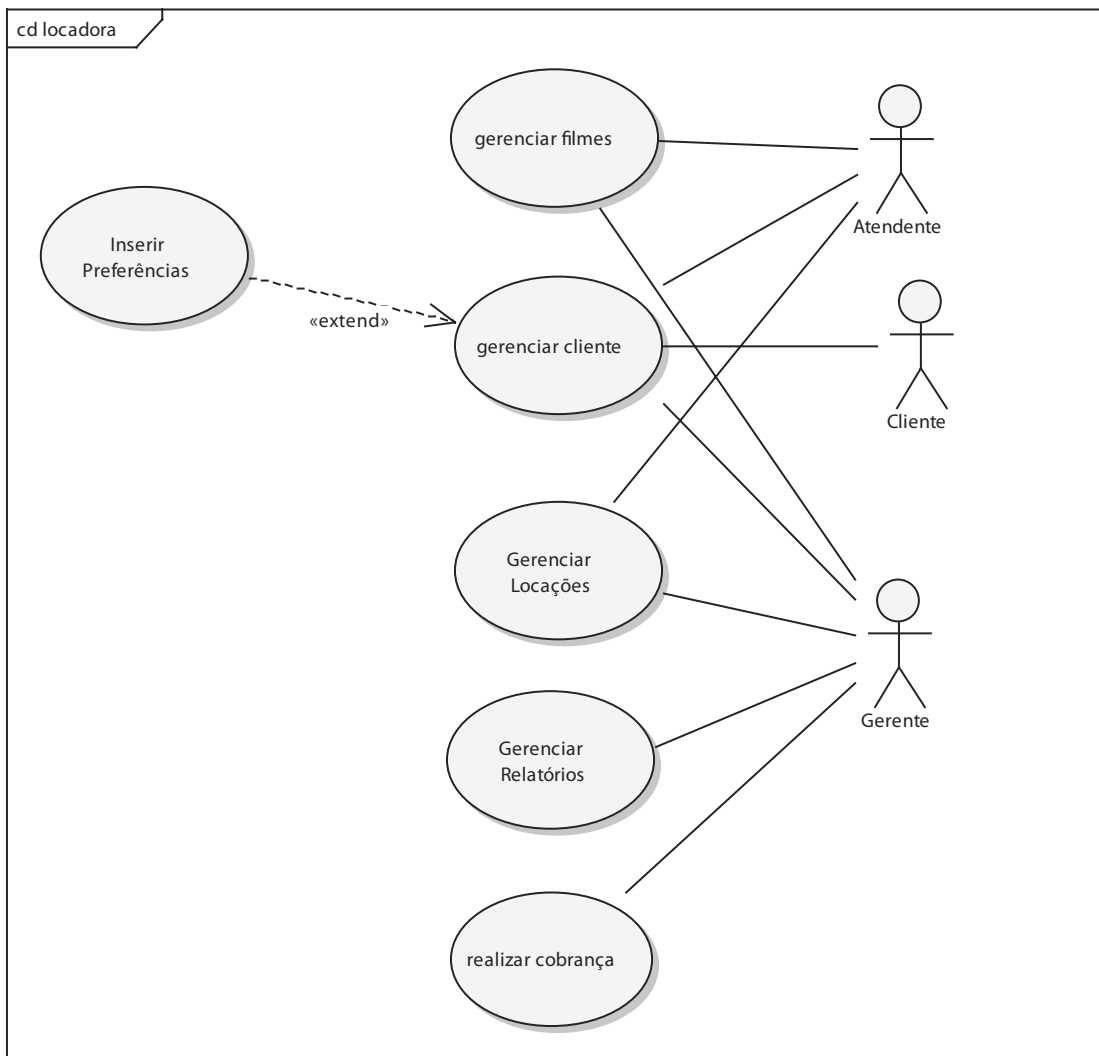



Figura 5.8. Diagrama de Casos de Uso da Vídeo Locadora

Utilize a extensão somente quando um comportamento opcional de um caso de uso precisar ser descrito.

Quando você faz uso da generalização significa que um caso de uso ou um ator herda características de um caso de uso ou um ator mais genérico.

Nestes casos existem características comuns que podem ser compartilhadas, um relacionamento entre um elemento geral e um outro mais específico. Neste caso o elemento mais específico possui todas as características do elemento geral e além destas contém ainda mais particularidades.

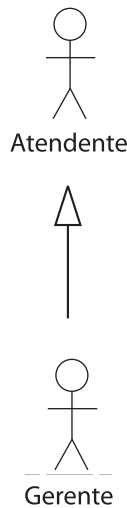



Figura 5.9 - Herança entre atores

Na generalização entre atores, como na figura acima, o ator gerente herda todos os casos de uso do ator atendente.

O ator atendente pode realizar os casos de uso gerenciar filmes, gerenciar cliente e gerenciar locações . O ator gerente herda todos estes casos de uso (pode realizar gerenciar filmes, cliente e locações), mas além destes, o ator gerente pode realizar o caso de uso gerenciar relatórios (que são específicos para este ator).

Se a generalização for entre casos de uso então podemos ter uma situação como a da figura a seguir (fig. 5.10).

Imagine uma funcionalidade no sistema da vídeo locadora onde será controlada a cobrança de clientes que sejam devedores, a cobrança pode ser realizada pessoalmente ou por telefone.

Nesta situação, o caso de uso Pessoalmente e Telefone, herda o comportamento do caso de uso realizar Cobrança, ou seja, além de herdar comportamentos básicos do caso realizar Cobrança, os dois casos possuem também fluxos de interações específicas para cada um deles.

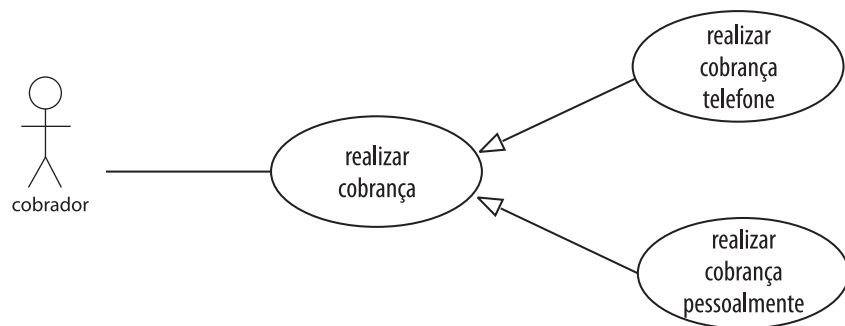



Figura 5.10 - Herança entre casos de uso

O diagrama geral de casos de uso atualizado seria então:

cd locadora

gerenciar filmes

Inserir Preferências

«extend»

gerenciar cliente

Gerenciar Locações

Gerenciar Relatórios

realizar cobrança

realizar cobrança pessoalmente

realizar cobrança telefone

Atendente

Cliente

Gerente

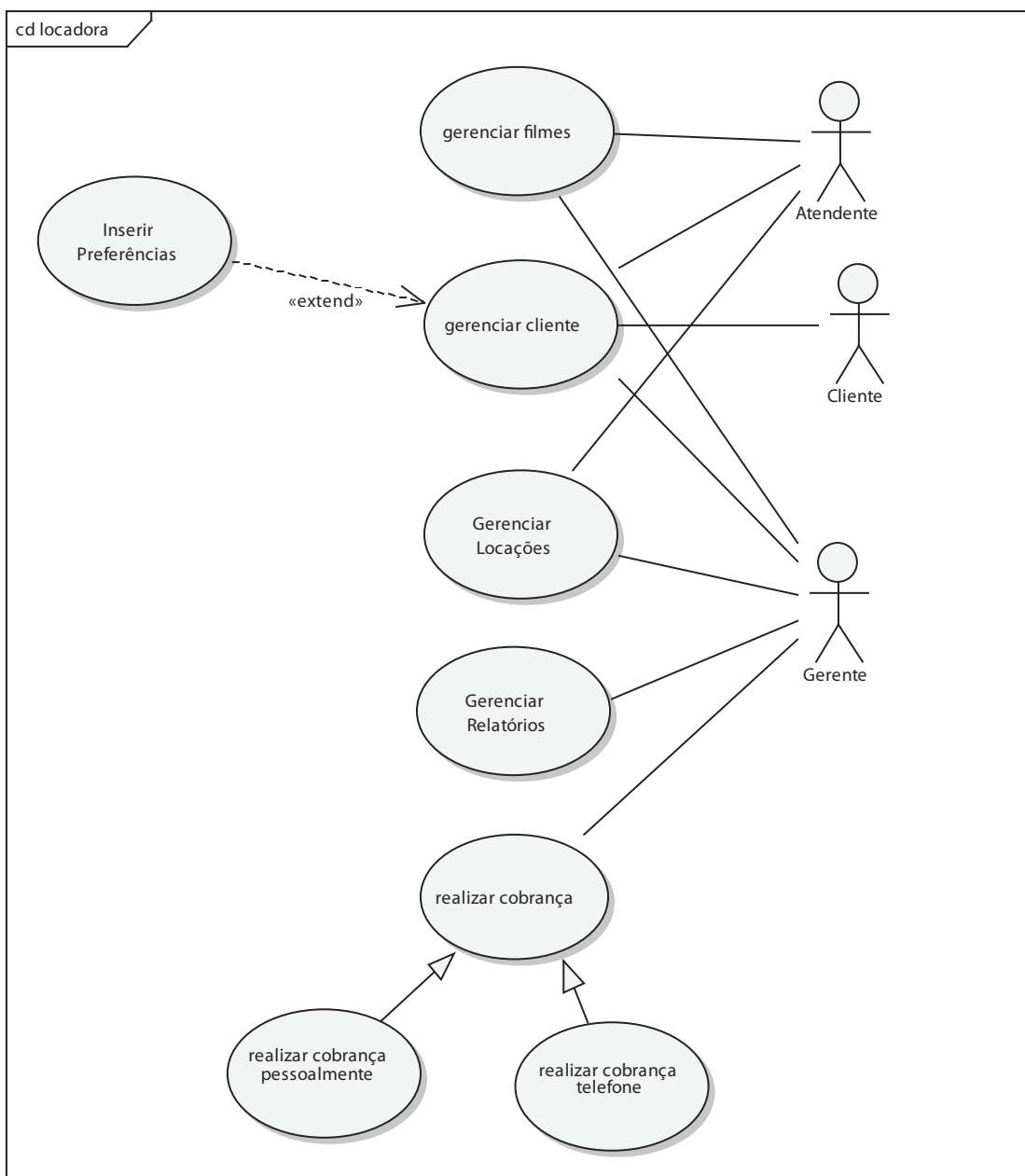



Figura 5.11 - Diagrama de Casos de Uso de Vídeo Locadora

O que é relacionamento de inclusão?

O caso de uso A inclui o caso de uso B quando representa uma atividade complexa e comum a vários casos de uso (PADUA, 2000). Este relacionamento só é possível entre casos de uso.

Em outras palavras o caso de uso pode ser incluído quando a seqüência de interações dele ocorre em mais de um caso de uso.

Ainda pensando no sistema da vídeo locadora observa-se que apenas o gerente tem acesso a cobrança e aos relatórios do futuro sistema. Para obter o acesso seguro o sistema deve apresentar uma funcionalidade que permita o controle do acesso por meio da autenticação do usuário no sistema . Esta autenticação será realizada para todos os casos de uso, verificando se o acesso está sendo realizado por um usuário credenciado a usar a função no sistema.

Nesta situação todos os casos de uso farão uso do caso de uso Autenticação, sendo um caso típico de inclusão, pois o mesmo caso de uso está relacionado a vários casos de uso.

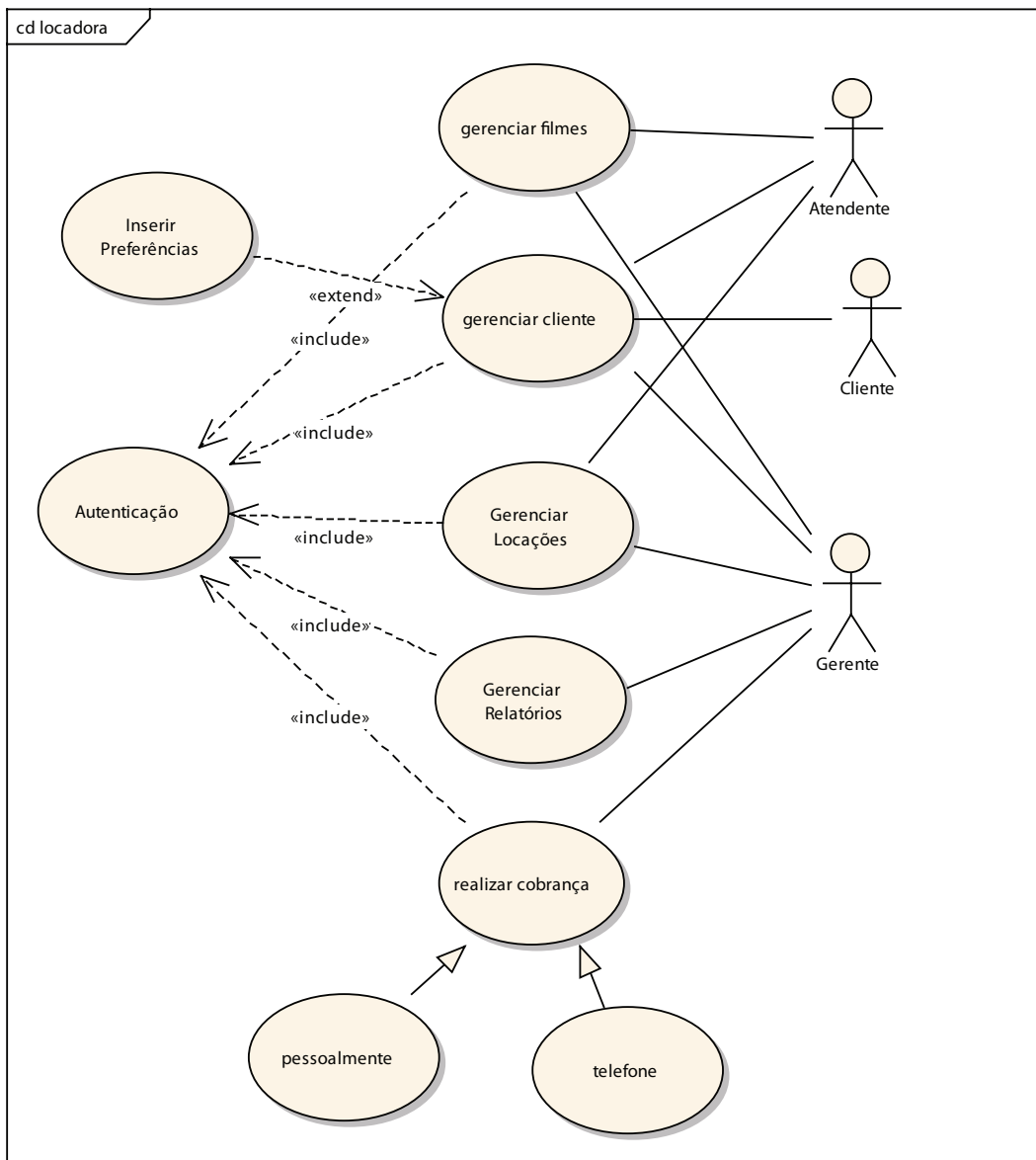



Figura 5.12. Diagrama de Casos de Uso da Vídeo Locadora

Um exemplo comum de inclusão é a autenticação por meio de contas e senhas. Quando você está em um caixa eletrônico com as possibilidades de saque, extrato, pagamento, apesar de cada uma delas possuir seqüências de interações e comportamento específico, têm em comum a autenticação de conta e senha. Logo, este caso pode ser incluído, pois em todos os processos ele vai acontecer com a mesma seqüência de interações.

Utilize o relacionamento de inclusão em situações onde o comportamento se repete em mais de um caso de uso.

SEÇÃO 4 - Identificando casos de uso

Mas, como identificar os casos de uso do sistema?

Identifique os objetivos do usuário e não-funções no sistema. Para identificar estes casos de uso PÁDUA (2000) sugere:

- Verifique quais as tarefas de cada ator
- Que informação cada ator cria, armazena, consulta, altera ou remove.
- Que informação cada caso de uso cria, armazena, consulta, altera ou remove.
- Que mudanças externas súbitas devem ser informadas ao produto pelos atores.
- Que ocorrências no produto devem ser informadas a algum ator.

Imagine uma situação onde você é convidado a desenvolver um projeto para um caixa eletrônico bancário. O projeto prevê o atendimento dos seguintes requisitos funcionais:

- O sistema deve permitir ao cliente a emissão de saldo somente da conta corrente;
- O sistema deve permitir ao cliente a emissão de extrato somente da conta corrente;

- O sistema deve permitir a atualização dos dados cadastrais do cliente;
- O sistema deve permitir o saque em dinheiro no caixa eletrônico;
- O sistema deve permitir a consulta a toda a movimentação financeira do cliente (conta corrente, poupança e aplicações) no caixa eletrônico;
- O acesso as funcionalidades do sistema deve ser possível somente após a verificação da conta e senha do cliente ou gerente.

Figura 5.13. Diagrama de Casos de Uso do Caixa Eletrônico

Como documentar o caso de uso?

Ao definir um caso de uso é necessária uma descrição narrativa das interações entre os elementos externos e o sistema.

E o que deve ser documentado para o caso de uso? Existem alguns aspectos que são fundamentais:

Tabela 5.2. Documentação do caso de uso

Campo	Descrição
Nome	O nome do caso de uso deve ser o mesmo nome que consta no diagrama. Não esqueça de que o nome de um caso de uso deve ser único.
Identificador	Convenção numérica utilizada para identificar o caso de uso. Exemplo : CSU002, CSU001
Descrição	Descrição sucinta do caso de uso
Ator Primário	O nome do ator que é o responsável pelo início do caso de uso.
Ator Secundário	Outros atores que fazem parte do caso de uso
Pré-condição	A pré-condição indica as condições necessários no sistema para que o caso de uso ocorra. Ex: Para realizar o caso de uso saldo, a pré-condição pode ser a validação positiva de conta e senha.
Fluxo Principal	O fluxo principal descreve a seqüência de ações que deve ocorrer quando o caso de uso é realizado.Seja breve na descrição, o fluxo deve ser escrito utilizando-se a terminologia do usuário.
Fluxo Alternativo	Descreve a seqüência de ações quando o ator faz uma escolha alternativa. O fluxo alternativo descreve um comportamento alternativo para o fluxo principal.
Pós-condição	Você deve descrever aqui o estado do sistema após a execução do caso de uso.
Regras de Negócio	Condições ou restrições na execução do caso de uso.

Fonte: Bezerra (2000).

A seguir, um exemplo do diagrama de caso de uso documentado:

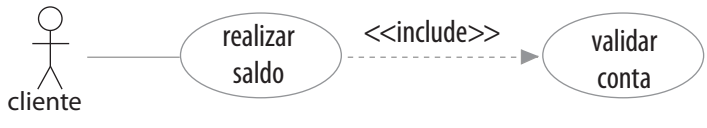
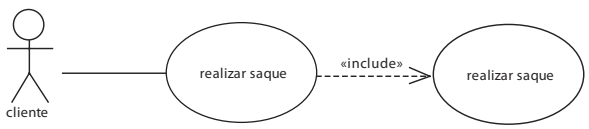
Tabela 5.3. Exemplo de Documentação do caso de uso realizar saldo

Nome	Realizar Saldo
Identificador	CSU001
Descrição	O cliente do banco solicita a emissão do saldo de sua conta corrente.
Ator Primário	Cliente
Pré-condição	Conta e senha do cliente serem válidas
Fluxo Principal	1. o cliente informa a conta e agencia 2. o cliente informa a senha 3. o sistema exibe a lista de serviços do caixa eletrônico 4. o cliente seleciona a opção saldo 5. o sistema apresenta mensagem informando a emissão do saldo
Fluxo Alternativo	1. O cliente cancela a operação por teclado finalizando o caso de uso
Pós-condição	Saldo do cliente impresso
Regras de negócio	RN01

Tabela 5.4. Exemplo de Documentação do caso de Uso Realizar Saque

Nome	Realizar Saque
Identificador	CSU003
Descrição	O cliente do banco solicita saque de sua conta corrente.
Ator Primário	Cliente
Pré-condição	Conta e senha do cliente serem válidas Haver saldo na conta superior ou igual ao valor solicitado
Fluxo Principal	1. o cliente informa a conta e agência 2. o cliente informa a senha 3. executa caso de uso "Validação Conta" 4. o sistema exibe a lista de serviços do caixa eletrônico 5. o cliente seleciona a opção saque 6. o cliente digita o valor a ser sacado 7. o cliente informa a senha 8. verifica se o saldo é suficiente 9. executa caso de uso "Validação Conta" 10. o valor do saldo da conta é atualizado 11. as cédulas são liberadas no ***dispenser*** de cédulas
Fluxo Alternativo 1	1. O cliente cancela a operação por teclado finalizando o caso de uso

Fluxo Alternativo 2	1. O sistema informa a mensagem “Conta e/ou Senha inconsistente” 2. O sistema finaliza a tarefa por senha ou conta inexistente
Fluxo Alternativo 3	1. O sistema informa a mensagem “Saldo Insuficiente para Saque” 2. O sistema finaliza a tarefa por saldo insuficiente
Pós-condição	Cédulas disponibilizadas
R. Negócio	RN02

Quais são as regras de negócio?

Quando você especifica uma regra de negócio você está especificando alguma condição, uma restrição, uma política ou uma norma que pode de alguma maneira interferir no processo que será realizado por seu projeto.

Regras de negócio mudam de uma empresa para outra, outras são comuns a várias empresas. Veja alguns exemplos de regras de negócio:

- no sistema de caixa eletrônico o cliente só poderá realizar o saque se e somente se: o saldo de sua conta corrente for superior ou igual ao valor a ser sacado. Se o saldo for inferior ao valor a ser sacado e se o cliente possui um valor de limite de crédito em sua conta corrente, então o saque não pode ser superior ao valor do saldo do valor limite da conta corrente.
- no sistema de estoque, na baixa do estoque quando a quantidade em estoque de um produto for inferior a quantidade de estoque mínimo deve ser gravada uma solicitação de pedido para este item. A quantidade solicitada será a diferença de quantidade sobre a quantidade mínima.

As regras de negócio podem ser documentadas por meio de uma identificação e descrição. Cada regra pode estar ligada a um ou mais casos de uso, neste caso o identificador deve ser anexado ao caso de uso.

Tabela 5.5. Exemplo de Documentação das Regras de Negócio

Identificador	Descrição da Regra de Negócio
RN01	O número máximo de impressão de saldos no mês é de 3 saldos mensais
RN02	O cliente só poderá realizar o saque se e somente se: o saldo de sua conta corrente for superior ou igual ao valor a ser sacado. Se o saldo for inferior ao valor a ser sacado e se o cliente possui um valor de limite de credito em sua conta corrente, então o saque não pode ser superior ao valor do saldo do valor limite da conta corrente.

Sugestão para Documentação dos Casos de uso

É importante estruturar a seqüência em que você vai organizar esta documentação. Existem metodologias que apóiam estas decisões como o RUP (unidade 9) e o Iconix. Uma sugestão de documentação pode obedecer esta seqüência:

- Se você não preencheu o documento de análise do problema e especificação de requisitos liste os requisitos funcionais e não funcionais.
- Documente o ator (lembra da tabela 1?)
- Insira o diagrama de casos de uso gerais do projeto.
- Uma tabela de documentação do caso de uso (tabela 3) para cada caso de uso.
- Documente as regras de negócio (tabela 4).

O uso de pacotes permite a visualização mais organizada principalmente em sistemas grandes.

Se houver muitos casos de uso ou atores, você pode usar os pacotes de casos de uso para estruturar ainda mais o modelo de casos de uso. Um pacote de casos de uso contém vários atores, casos de uso, seus relacionamentos e outros pacotes.

Imagine que você tenha a seguinte lista de casos de uso:

- Cadastro de Clientes;
- Pedidos em Aberto;
- Gerenciamento de Vendas;
- Relatório de Data de Validade Vencidas;
- Cadastro de Produtos;
- Lista de Clientes;
- Lista de Produtos;
- Relatório de Produtos Mais Vendidos;
- Pedidos de Cancelamento.

Iconix - O ICONIX é um processo simplificado que unifica conjuntos de métodos de orientação a objetos em uma abordagem completa, com o objetivo de dar cobertura ao ciclo de vida do desenvolvimento de *software*. (ROSENBERG & SCOTT, 1999).

RUP – Rational Unified Process é definido como um processo de engenharia de software que oferece uma abordagem baseada em disciplinas possibilitando a atribuição de tarefas e responsabilidades no desenvolvimento de software (Souza et all, 2000).

O projetista pode agrupar casos de uso que apresentam aspectos comuns como o tipo de função, atores que utilizam ou questões organizacionais. Observe na figura a seguir, os casos de uso foram divididos em três pacotes, este agrupamento facilita a visualização principalmente em sistemas com um grande número de casos de uso.

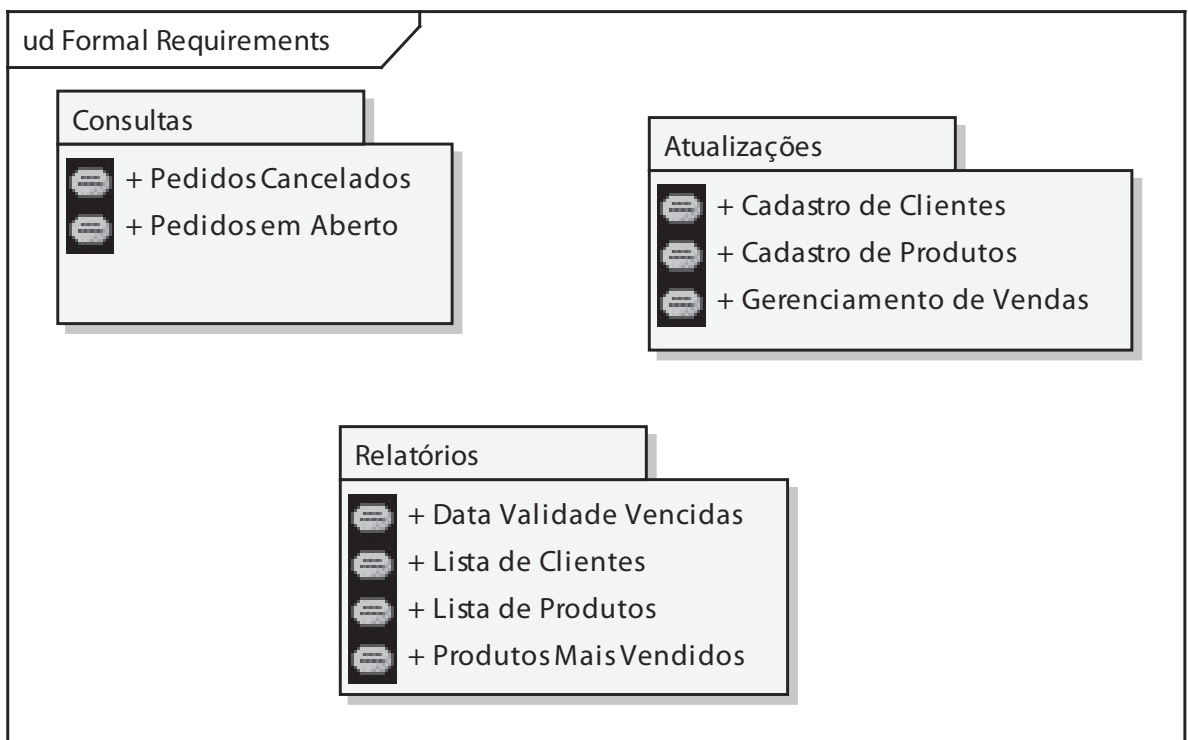



Figura 5.14. Diagrama de Pacotes

Quer conhecer mais ?

Para melhorar seus conhecimentos sobre esta unidade você pode ler:

- O capítulo 4 Princípios de Análise e Projeto de Sistemas com UML, BEZERRA, E., CAMPUS, 2002
- Na midiateca você vai achar dois artigos interessantes abordando o assunto sob os links:
  UML - Linguagem de Modelagem Unificada
  – Diagrama de caso de uso

Que tal acompanhar mais um exemplo?

Você foi contratado para modelar um sistema de imobiliária. Após a análise os requisitos funcionais foram assim definidos:

RF01 -> Requisito Funcional + numeração seqüencial

Cadastro de clientes	RF001
Permitir a inclusão, alteração e exclusão de clientes para compra, venda ou aluguel de imóveis.	

Cadastro de corretores	RF002
Permitir o cadastro de corretores que vendem imóveis para a imobiliária.	

Cadastro de Fiador	RF003
Cadastro dos fiadores usados pelos clientes permitindo Incluir, Alterar, Excluir seus dados.	

Cadastro de imóveis	RF004
Cadastra todos dados dos imóveis que são negociados na imobiliaria . Imóveis podem ser para locação ou venda: casa, ap, kit net, comercial.	

Alugar imóvel	RF005
Cadastra os dados de aluguel de um imóvel para um cliente como data de locação, data de término de contrato, valor do aluguel.	

Vender Imóvel	RF006
Cadastro das vendas realizadas pelos corretores permitindo incluir, alterar e excluir registros de venda.	

Gerar Contrato	RF007
Gera o contrato para ser impresso no ato da venda ou aluguel do imóvel. A geração do contrato e dados deve ser automática.	

Emitir boleto bancário	RF008
Emite o boleto de cobrança para clientes que alugam imóveis com dados como valor do aluguel, iptu, descontos e multas.	

Controle das comissões	RF009
Emite um extrato com os imóveis alugados ou vendidos indicando o valor da comissão do corretor.	

Busca de imóvel	RF010
Propiciar a realização da busca de imóvel por parâmetros informados como bairro, número de quartos, valor aproximado de aluguel.	

Gerar Relatório de Cobrança	RF011
Permitir a geração de um relatório com os imóveis alugados que estão com mensalidade atrasada.	

Cadastra Pagamento	RF012
Deve ser permitido o registro do pagamento de aluguel de um imóvel.	

Gerar Relatório	RF013
Gerar um relatório com quantidade de vendas e alugueis realizados e desfeitos. Classificado por mês e ano.	

Cadastrar Usuário do Sistema	RF014
Permitir o cadastro de usuários do sistema, para que se possam definir níveis de acesso por meio de contas e senhas.	

Login de Usuário	RF015
Efetuar login para identificar quem está usando o sistema e definir os acessos que ele possui	
Cadastro de Manutenções	RF016
Cadastra dados de manutenções feitas no imóvel armazenando o tipo de manutenção, o valor gasto, data da manutenção e uma breve descrição .	

Requisitos Não Funcionais

RNF01 -> Requisito Não Funcional + numeração seqüencial

Tempo de resposta	RNF01
O tempo de resposta para consultas ao sistema, como a busca de imóvel, não devem ser inferiores a 5 segundos.	

Manutenibilidade	RNF02
O Sistema deve ser construído obedecendo a visão de camadas facilitando futuras manutenções.	

Durante a análise perceberam-se as regras de negócio relacionadas ao funcionamento do processo na imobiliária:

Tabela 5.6. Regras de Negócio Sistema Imobiliária

Identificador	Descrição
RN01	O cliente deve ter fiador para contratar aluguel
RN02	CPF do Cliente e Fiador devem ser válidos.
RN03	O imóvel deve ser aprovado pelo gerente antes de ser cadastrado.
RN04	O crédito do cliente deve ser aprovado pelo gerente antes de efetivar o contrato de aluguel.
RN05	O valor da taxa do boleto, cobrada pelo banco, é adicionado no valor total do boleto do locador.

RN06	A multa por atraso no pagamento do aluguel é de 0,1% do valor do aluguel por dia de atraso.
RN07	O pagamento do boleto após o vencimento só pode ser efetivado em estabelecimento bancário conveniado a imobiliária.
RN08	As manutenções ou reformas feitas pelo locatário no imóvel, são por conta própria, ou seja, não serão abatidas no valor da mensalidade.

No sistema imobiliário foram identificados 4 atores:

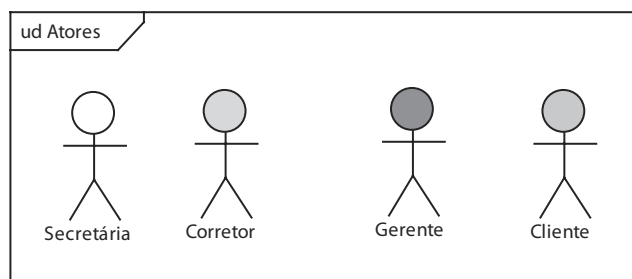



Figura 5.15. Atores do Sistema Imobiliário

A partir dos requisitos funcionais foram definidos os casos de uso. Os casos de uso foram subdivididos em 3 pacotes Negociar Imóvel, Gerenciamento e Administração. Os casos de uso relacionados a cada pacote encontram-se inseridos no mesmo.

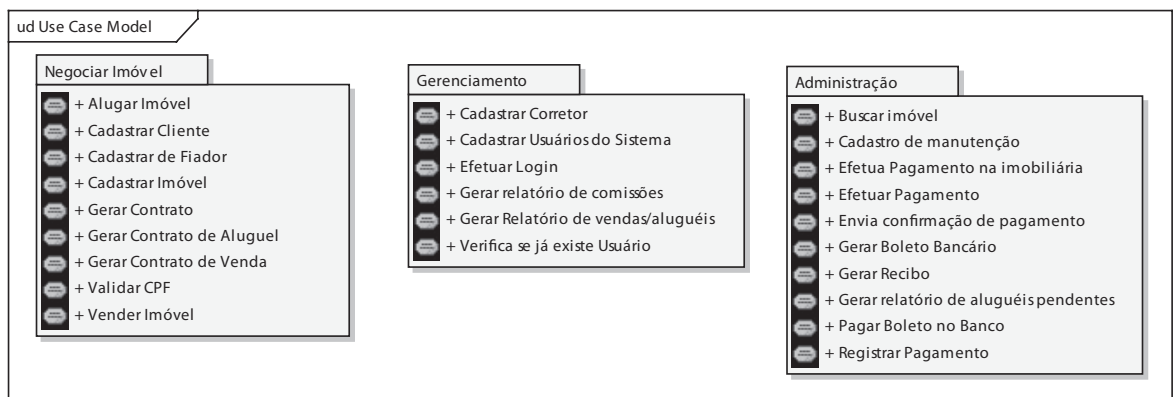



Figura 5.16. Pacotes de Casos de Uso do Sistema Imobiliário

O diagrama de casos de uso pode ser feito de forma geral ou por pacotes, observe a seguir diagrama de casos de uso dos pacotes Gerenciamento e Negociar Imóvel:

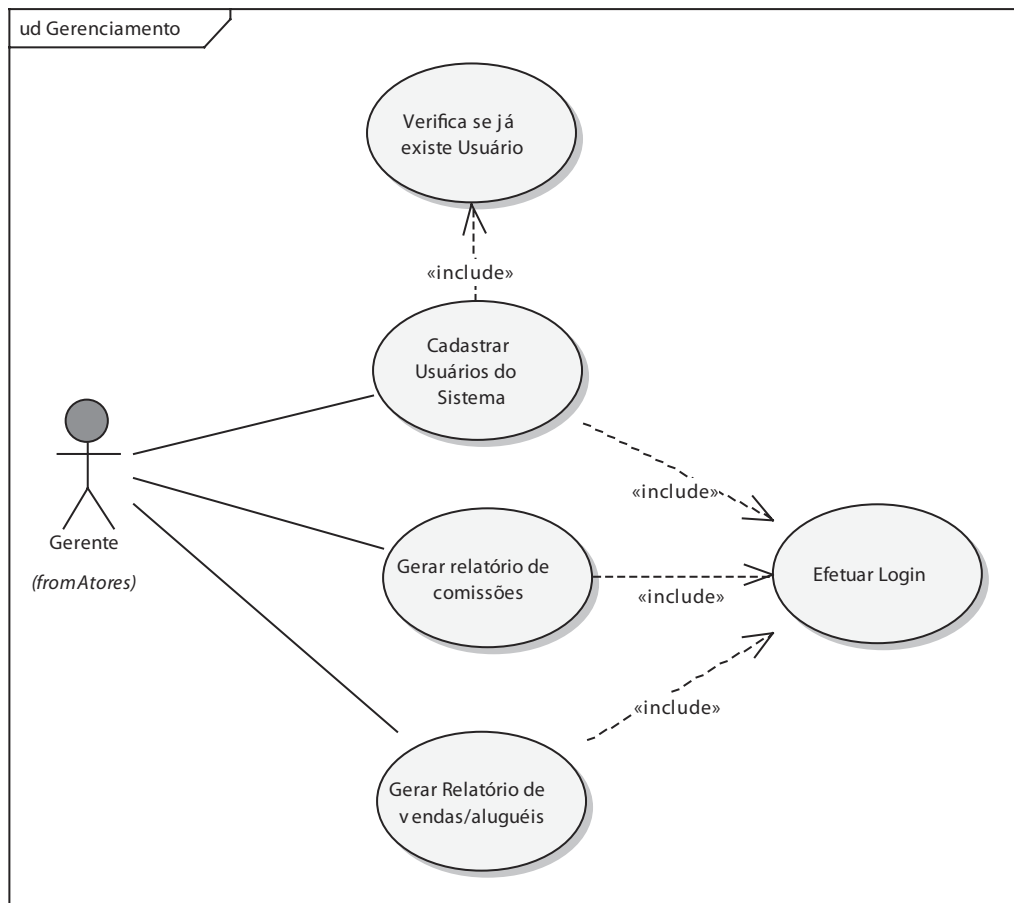



Figura 5.17. Diagrama de Casos de Uso do Pacote Gerenciamento

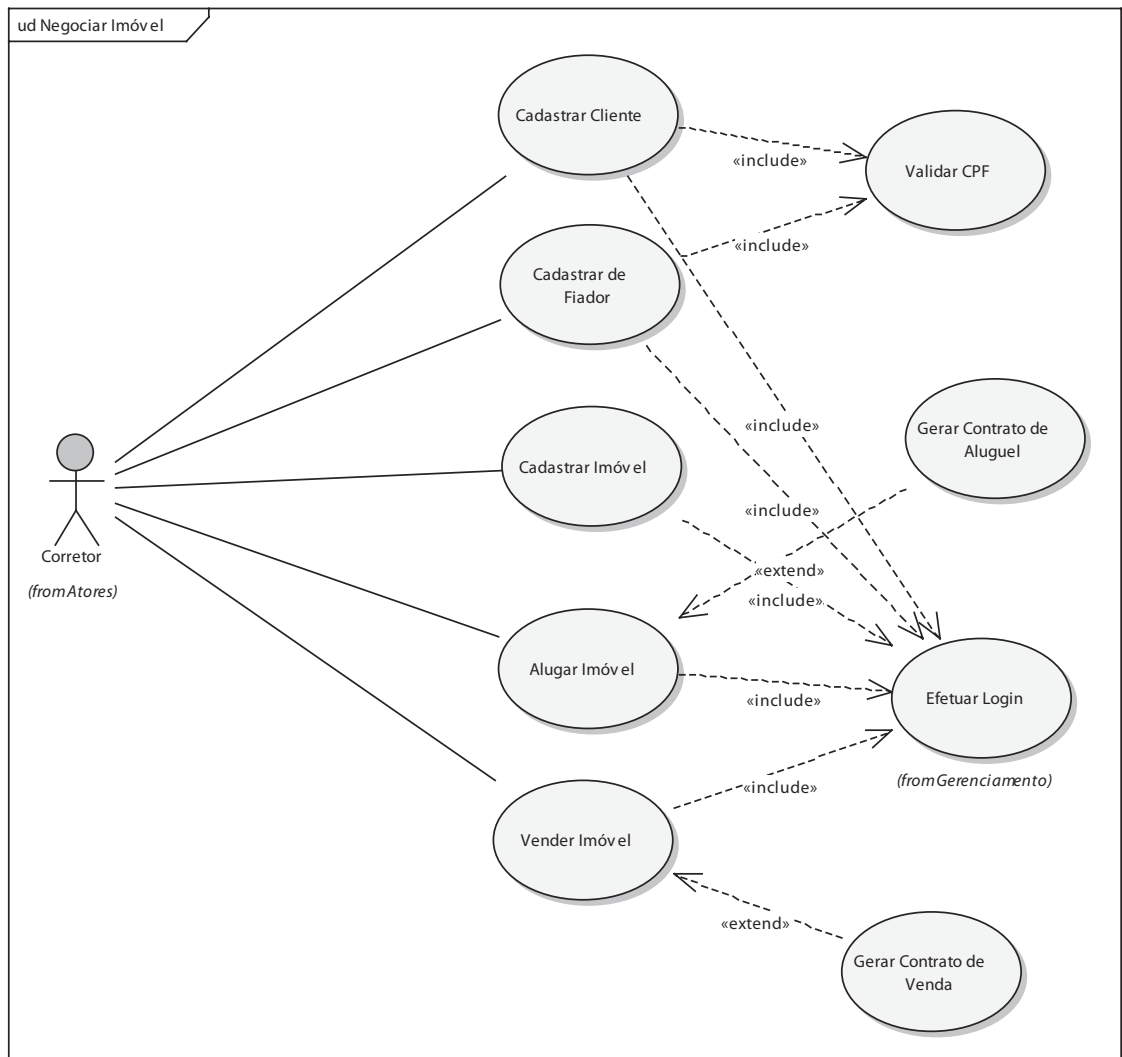



Figura 5.18. Diagrama de Casos de Uso do Pacote Negociar Imóvel

Definidos os diagramas você vai documentar cada caso de uso, como os exemplos apresentados a seguir:

Tabela 5.7. Documentação do caso de uso Cadastrar Fiador

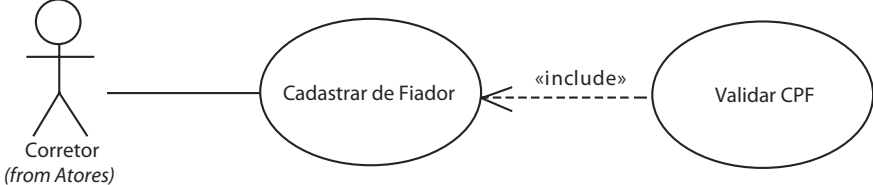
Corretor
(from Atores)
Cadastrar de Fiador
«include»
Validar CPF

Nome	Cadastrar Fiador
Identificador	CSU02
Descrição	O corretor cadastra as informações inerentes ao fiador de um cliente.
Ator Primário	Corretor
Pré-condição	Corretor logado no sistema
Fluxo Principal	1. Corretor digita dados do fiador no sistema. 3. Corretor Confirma cadastro. 4. Executa "Validar CPF". 5. Dados do Fiador são armazenados.
Fluxo Alternativo 1	1. Corretor seleciona botão Cancelar. 2. Os dados da janela são limpos.
Fluxo Alternativo 2	1. Corretor deseja alterar dados de algum fiador 2. Corretor seleciona um fiador cadastrado. 3. Sistema mostra informações do fiador. 4. Corretor altera informação 5. Executa "Validar CPF". 6. Corretor confirma a alteração 7. Dados do Fiador são armazenados
Fluxo Alternativo 3	1. Corretor deseja excluir fiador. 2. Corretor seleciona um fiador cadastrado. 3. Seleciona botão Exclusão. 4. Dados do fiador são excluídos do banco de dados.
Pós-condição	Dados do fiador atualizados
Regras de Negócio	RN02

Tabela 5.8 - Documentação do caso de uso Cadastrar Imóvel

Corretor *(from Atores)*	Cadastrar Imóvel
Nome	Cadastrar Imóvel
Identificador	CSU04
Descrição	Efetua o cadastro do imóvel que ficará disponível na imobiliária.
Ator Primário	Corretor
Pré-condição	Corretor logado no sistema e o Proprietário do imóvel precisa estar cadastrado no sistema como cliente.
Fluxo Principal	1. Corretor informa código do cliente 2. Corretor insere dados do imóvel no sistema. 3. Corretor confirma cadastro.
Fluxo Alternativo 1	1. Corretor seleciona botão Cancelar. 2. Os dados da janela são limpos.
Fluxo Alternativo 2	8. Corretor deseja alterar dados do imóvel 9. Corretor seleciona um imóvel cadastrado. 10. Sistema mostra informações do imóvel. 11. Corretor altera informação 12. Corretor confirma a alteração 13. Dados do Imóvel são armazenados
Fluxo Alternativo 3	5. Corretor deseja excluir imóvel. 6. Corretor seleciona um imóvel cadastrado. 1. Seleciona botão Exclusão. 2. Dados do imóvel são excluídos do banco de dados.
Pós-condição	Imóvel atualizado no banco de dados.
Regras de Negócio	RN03

Agora para praticar os conhecimentos conquistados nesta unidade, realize a seguir as atividades propostas

Atividades de auto-avaliação

Leia com atenção os enunciados e, após, realize as questões propostas:

1) Assinale a afirmativa correta:

a) ( ) Um caso de uso procura apoiar a especificação de detalhes necessários à implementação do sistema.

b) ( ) O caso de uso documenta as ações necessárias, comportamentos e seqüências visando atender as necessidades do usuário.

c) ( ) Em um caso de uso são necessários três elementos básicos: o caso de uso, a classe de controle e o ator.

d) ( ) O ator de um caso de uso representa apenas os usuários do sistema.

2**)** Classifique as questões em Verdadeira (V) ou Falsa (F).

a) ( ) Um ator pode ser representado por um organização ou mesmo um equipamento de *hardware*.

b) ( ) O uso de um nome pessoal para um determinado ator cria uma identificação pessoal do usuário com a especificação do caso de uso, facilitando sua validação com o usuário.

c) ( ) A relação existente entre um caso de uso e um ator pode ser uma relação de comunicação ou uma relação de extensão.

d) ( ) A relação existente entre dois atores pode ser somente uma relação de herança.

e) ( ) A relação existente entre dois casos de uso pode ser de extensão, inclusão e comunicação.

f) ( ) A relação existente entre dois casos de uso pode ser de extensão, inclusão e herança.

3) Relacione a 1ª com a 2ª coluna.

A. Regras de negócio	a) ( ) Fitas em atraso pagam o valor da locação dos dias atrasados
B. Ator	b) ( ) Atendente
C. Cenário	c) ( ) "A reserva de uma fita pode ser realizada pessoalmente onde o cliente informa seu nome, informa a fita desejada e a data da reserva. Caso a fita não esteja reservada o atendente realiza a reserva. A fita pode ser reservada na página da videolocadora, informando seu nome o cliente seleciona a fita desejada e informa a data da reserva. A reserva é efetiva somente pelo envio da confirmação para o e-mail do cliente."
D. Caso de uso	d) ( ) Devolução de Fitas
	e) ( ) Fornecedor de fitas
	f) ( ) Gerenciar compra de fitas
	g) ( ) O cliente pode retirar no máximo 3 fitas com devolução para 24 horas.

4) No texto abaixo, identifique:

a) Os requisitos funcionais

b) Os atores

c) Os casos de uso

d) Documente 2 casos de uso

Requisitos funcionais:

- O projeto que você vai realizar será para a clinica pediátrica Bem-Estar e tem como objetivo principal a marcação de consultas médicas.
- O paciente pode realizar o agendamento da consulta pessoalmente ou por telefone, em qualquer dois dos métodos os procedimentos são idênticos.
- O paciente solicita a consulta informando o nome do médico ou a especialidade desejada, posteriormente informa a data desejada. A atendente verifica a possibilidade de marcação da consulta (observando se o médico em questão possui horários vago para a data desejada). Se existe horário disponível a atendente solicita ao paciente o tipo de convenio ou se é particular. Se for convênio é verificado se é um convênio válido, se for particular é informado o valor da consulta. A atendente atualiza a agenda com o nome do paciente e o tipo de

consulta(convênio/particular). O tempo para cada consulta é de 20 minutos ou 10 minutos para retorno.

- A consulta pode ser uma consulta de retorno, neste caso a atendente verifica se a data está ainda dentro do prazo de retorno de 15 dias. Em caso a consulta é marcada na agenda.
- Caso o médico solicitado esteja indisponível a atendente procura informar o nome de outro médico disponível naquele horário ou o próximo horário disponível.
- O paciente pode telefonar desmarcando a consulta, neste caso o nome do paciente é riscado da agenda ficando o horário vago novamente.
- Os médicos ao chegarem na clínica recebem as fichas dos pacientes separadas previamente. Se for paciente novo a ficha contém somente o nome do paciente e o telefone. As fichas são classificadas por ordem de horário.
- Se o paciente já possui cadastro o mesmo é convidado a adentrar no consultório do médico. A partir deste momento o médico solicita informações procedimentais para o futuro diagnóstivo, preenchendo a ficha do paciente. Finalizada a consulta o paciente é liberado e a ficha é recolhida pela atendente sendo novamente arquivada.
- Se o paciente for novo a atendente solicita o preenchimento da ficha cadastral com dados pessoais.

Síntese

Nesta unidade você aprendeu a identificar os casos de uso mais significantes e seus principais elementos.

Durante o estudo também foram identificados os possíveis relacionamentos existentes entre atores e casos de uso. Você percebeu que o caso de uso estabelece a estrutura principal da modelagem permitindo uma comunicação fácil e precisa com o usuário, pois esta visão não inclui aspectos relacionados à implementação facilitando a interpretação por parte de desenvolvedor e usuários, identificando as ações necessárias ao sistema para que o usuário alcance a solução de suas necessidades.

A partir dos conceitos elementares você seguiu em direção à concepção do diagrama e da documentação necessária para o bom entendimento do modelo.

Na próxima unidade você vai estudar a composição dos diagramas de classes que fazem parte da visão de projeto do sistema. Você lembra o que é uma classe ? Uma classe representa uma coleção de objetos com propriedades, comportamento e relacionamentos comuns com outros objetos.

Saiba mais

Para aprofundar as questões abordadas nesta unidade você poderá pesquisar :

BOOCH, G.,RUMBAUGH, J., JACOBSON, I., **UML** Guia do Usuário. Ed. Campus, 2000. (capítulo 16 e 17)

UNIDADE 6

6

Modelagem de classes

Objetivos de aprendizagem

- Identificar o papel do diagrama de classes no processo de análise.
- Conhecer e reconhecer termos técnicos, conceitos e relacionamentos utilizados durante a construção do diagrama de classes.
- Identificar as possíveis classes de um projeto.
- Utilizar o diagrama de classe com a notação UML na construção de um modelo de projeto.

Seções de estudo

Seção 1 O que são objetos e classes de objetos?

Seção 2 Quais são as responsabilidades das classes?

Seção 3 Como ocorrem os relacionamentos entre objetos?

Seção 4 Como ocorre a divisão das classes do Modelo de Análise?

Seção 5 O que é diagrama de objetos?

Para início de estudo

Na unidade anterior você estudou um dos aspectos mais fortes do projeto, que são os casos de uso. Os casos de uso permitem aos atores a visualização de resultados esperados, relatórios e processamentos.

De qualquer maneira, a possibilidade da existência desta colaboração depende de aspectos estáticos e dinâmicos do sistema. Entre objetos do sistema, o aspecto dinâmico está fortemente ligado à troca de mensagens, enquanto que o aspecto estático mostra como o sistema "está estruturado internamente" e passa por três níveis de abstração:

- o primeiro deles, o modelo de classe de domínio, representa a classe de domínio sem se preocupar com detalhes sobre a tecnologia;
- o segundo nível é o modelo da classe de especificação que acrescenta detalhes relacionados à solução do *software* escolhida pelo modelo do domínio;
- o terceiro nível, o modelo de classe de implementação, é uma extensão do modelo de especificação, ocorre neste nível a implementação em alguma linguagem de programação.

Para você entender a utilização destes níveis de abstração é necessário conhecer conceitos e relacionamentos vinculados ao modelo de classes. O modelo de classes é um dos modelos mais ricos em termos de notação e concentra o cerne estático de todo o projeto.

Então, que tal escalar o mundo conceitual das classes?

SEÇÃO 1 - O que são objetos e classes de objetos?

De acordo com PÁDUA (2000), as entidades de domínio são representadas na modelagem orientada a objetos por objetos. O objeto representa uma entidade que pode ser física ou de *software*.

Na realidade, um objeto sempre é descrito por meio do estado, comportamento e identidade:

- A identidade é uma propriedade que irá distinguí-lo dos demais.
- O estado de um objeto compreende características herdadas ou distintas que contribuem para que se torne único.
- O comportamento de um objeto define como sua ação e reação a estímulos em termos de mudanças de estados e mensagens.

Classe Cliente

Identidade
Cliente

Estado
Nome
Código
Endereço
Telefone
Permissão

Comportamento:
Adicionar Cliente( )
Excluir Cliente( )
Consultar Cliente( )

Você pode dizer também que, **um objeto é uma pessoa, um lugar ou um sistema**, como mostra a figura a seguir.

Os objetos também são chamados de instância.

Figura 6.1 - Exemplos de Objetos

Segundo FURLAN (1998), **classe é a representação de um conjunto de coisas reais ou abstratas que são reconhecidas como sendo do mesmo tipo por compartilhar as mesmas características de atributos, operações, relações e semântica.**

Um objeto é sempre uma instância de uma classe. Quando você fala de uma classe está falando também de seus objetos.

BOOCH (2000) define classe como um conjunto de objetos que compartilham estrutura e comportamento comuns.

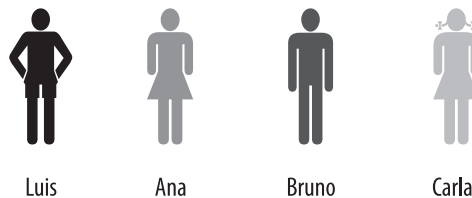



Figura 6.2 - Classe Clientes

Conforme você pode observar na figura 2, Luis, Ana, Bruno e Carla são objetos ou instâncias da classe. Mas, qual a notação utilizada para identificar uma classe?

A classe é representada por um retângulo dividido em três compartimentos que contém o nome da classe, os atributos e as operações.

Figura 6.3 - Notação da Classe

O atributo é a descrição dos dados armazenados pelos objetos de uma classe. O atributo de uma classe está associado a um conjunto de valores que o atributo pode assumir.

Está correto afirmar que, **os atributos não têm comportamento**. Cada valor de um atributo é particular para um dado objeto.

Uma classe pode ter qualquer número de atributos ou mesmo nenhum atributo. Os atributos são sempre individuais e cada objeto da classe possui seus próprios atributos.

Recordando do projeto da videolocadora, você pode definir as classes?

É possível detectar imediatamente três classes neste projeto :

- Classe Cliente (Dados e métodos do cliente)
- Classe Filmes (Dados e métodos dos filmes)
- Classe Locação (Dados e métodos sobre a locação)

As operações implementam serviços que podem ser solicitados por algum objeto da classe para modificar o comportamento, ou seja, todos os objetos da classe vão compartilhar destas operações.

As operações são executadas quando um objeto recebe uma mensagem de outro objeto. Entretanto, existem situações onde uma classe pode não ter nenhuma operação ou mesmo ter várias operações.

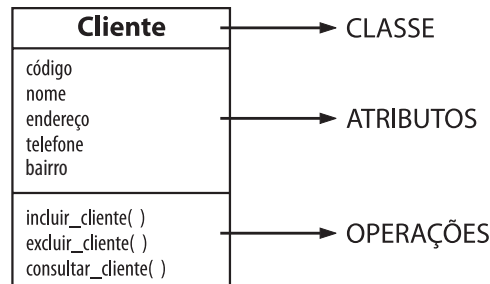



Figura 6.4 - Classe Cliente

Observe que:

- **para nomear a classe** você deve escolher um substantivo, por exemplo: Fornecedor, Produtos, Cliente;
- **para nomear uma operação** faça uso de um verbo ou mesmo de um verbo mais um substantivo. A escolha do nome da operação deve possuir um nome que indique o resultado da operação (Cancelar_Fornecedor) acrescentando ao final, os parênteses "( )" ao final do nome da operação;
- **para nomear atributos** utilize-se de substantivos simples ou verbos substantivados. Se você for construir uma classe para um sistema de controle de estoque chamada Produtos. Os atributos da classe produto podem ser código, descrição_Produto, unidade;
- para os três casos ao atribuir nomes utilize uma definição concisa e clara, não dando margem a interpretações errôneas.

Quais seriam os atributos para a classe Filmes?

Você poderia ter neste caso:

- Nome_
- Código_
- Diretor_
- Duração_
- Ator_Principal1
- Ator_Principal2
- Tipo_
- Idiomas_

Agora tente imaginar as possíveis operações:

- Incluir_Filme
- Excluir_Filme
- Consultar_Filme
- Listar_Filme

Observe algumas classes que fariam parte do domínio do Sistema da Vídeo Locadora.

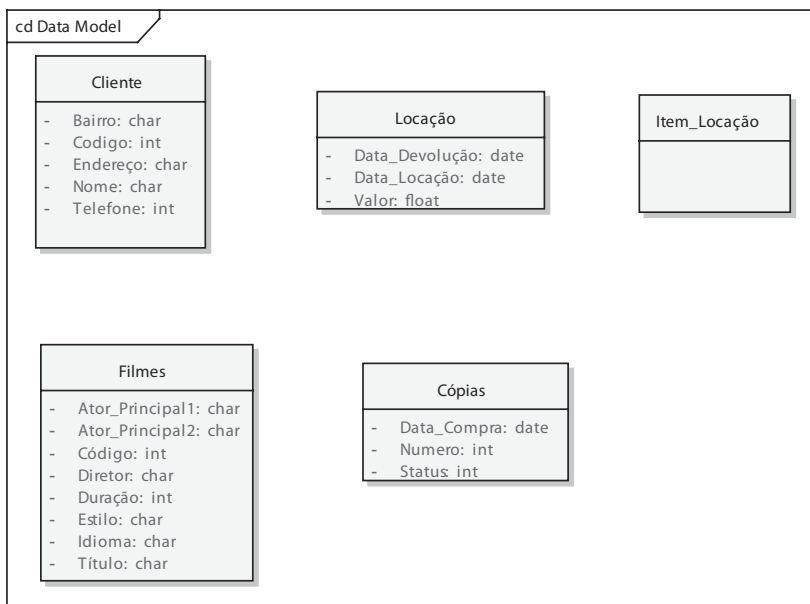



Figura 6.5 - Possíveis Classes do SIstema de Vídeo Locadora.

A notação de classes apresenta 3 compartimentos. Porém as classes podem ser apresentadas em diferentes níveis de abstração como apresentado na figura abaixo:

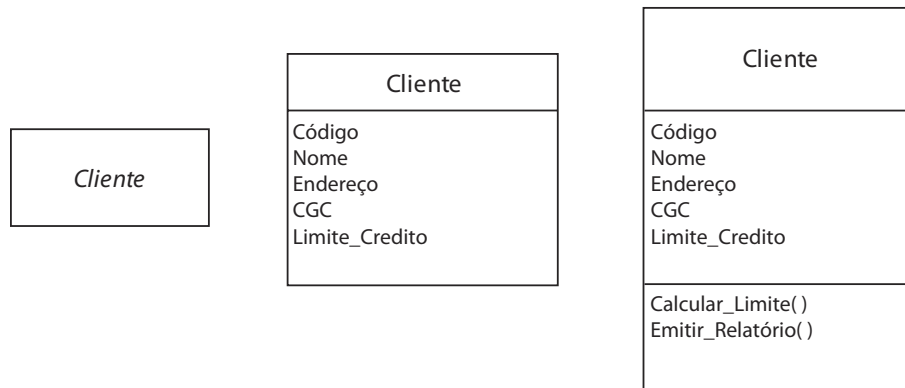



Figura 6.6. Abstrações das Classes

SEÇÃO 2 - Quais são as responsabilidades das classes?

Uma responsabilidade é um contrato ou obrigações de uma determinada classe, ou seja, **representam o conhecimento e ações que possibilitam as classes cumprir seu papel nos casos de uso**.

Por exemplo: uma classe cliente é responsável pelo conhecimento sobre os dados pessoais (código, nome, endereço...) do cliente, além de ser responsável por incluir, excluir e consultar os dados do cliente.

Mas, definir as classes existentes em um futuro sistema e suas responsabilidades não é uma tarefa fácil. O mais adequado é você procurar o apoio de métodos reconhecidos que procuram facilitar sua realização. Para isso, existem dois métodos popularizados para esta etapa que são: os cartões CRC e a análise dos casos de uso.

a) Os cartões CRC - Classes Responsabilidades e Colaborações

O uso dos cartões CRC identifica as responsabilidades dos atributos e das operações apoiando a identificação de classes ou de candidatos às classes.

Cada ficha corresponde a uma classe, cada ficha contém o nome da classe e 2 colunas com descrição de suas responsabilidades e colaborações. As colaborações representam outras classes que interagem com a classe descrita para o cumprimento de suas responsabilidades.

O uso do cartão CRC é realizado com o envolvimento de toda a equipe e para isso uma sessão é organizada. Para realizar a sessão:

O primeiro passo é a escolha do grupo de pessoas que irá representar um cenário, ou seja, é exatamente o cenário do domínio do problema;

Para cada Classe de objeto identificada dentro do cenário é criado um cartão CRC.

- Na segunda etapa cada participante é associado a uma classe. Assim, cada pessoa passa a pertencer aquela classe e todo o cenário será encenado pelos participantes. Aos poucos cada cartão é preenchido com as responsabilidades e os colaboradores.

Durante a sessão pela exploração dos cenários é comum que novos cartões sejam criados pela descoberta de novas classes.

Observe o cartão CRC criado a partir do seguinte cenário:

O balconista faz a abertura da venda.

O balconista registra os itens venda podendo inserir novos itens de venda, excluí-los e editá-los.

O sistema totaliza a venda para o cliente.

O sistema calcula os impostos sobre a venda.

O balconista encerra a venda.

Tabela 6.1 - Exemplo de um cartão CRC

Nome da Classe: Venda	
Responsabilidades	**Colaborações**
Inserir/excluir item de venda	item de venda
Editar item de venda	Estoque
Totalizar venda	Mercadoria
Calcular impostos	Mercadoria

Você pode documentar as responsabilidades no próprio diagrama de classes. Neste caso insira a descrição na forma de texto no final da caixa da classe, ou seja, na notação gráfica você tem mais um compartimento logo abaixo das operações.

b) Análise dos casos de uso

Se você optar por utilizar a análise de casos de uso para identificar as classes candidatas e suas responsabilidades, é importante analisar os casos de uso e todos os seus fluxos (principais e alternativos).

Mas, como funciona este método? Bom, a partir da análise são identificados os substantivos existentes nos casos de uso e os sinônimos são eliminados.

Muitas vezes, um substantivo pode ser um ator, o qual deve ser eliminado. Assim, os nomes que permaneceram são as classes candidatas.

Observe o exemplo a seguir:

Pádua (2000) mostra a análise de um estudo de caso para um sistema de vendas de uma pequena mercearia. Ao analisar o estudo de caso procurando as classes candidatas são selecionados em itálico os atores, o sistema aparece em negrito e em sublinhado os substantivos.

O *balconista* faz a abertura da venda.

O *balconista* registra os itens vendidos informando código do produto e quantidade do item.

O **sistema** totaliza a venda para o cliente.

O *balconista* encerra a venda.

O **sistema** emite o ticket para o cliente.

O *balconista* registra a forma de pagamento.

O **sistema** faz a baixa no estoque das mercadorias vendidas.

Ao analisar o caso de uso, você deduz a possível lista de classes candidatas, operações e atributos:

Tabela 6.2 - Exemplo lista de classes

Classe Candidata	Análise
Abertura	Operação
Venda	Provável classe
Item vendido	Provável classe – melhor item de venda
Código do produto	Atributo do item da venda
Quantidade	Atributo do item da venda
Cliente da mercearia	Entidade fora do escopo do produto
Ticket	Relatório
Forma de pagamento	Atributo da venda
Baixa	Operação
Estoque	Possível classe
Mercadoria	Possível classe

Observe que o item **abertura** pode ser descrito como uma operação, assim como **venda** passa a ser uma possível classe, já

código do produto tem as características de um atributo. Note que já é possível identificar inclusive a qual classe o atributo pertence como no caso da **quantidade**.

A partir da identificação é possível determinar no exemplo as classes candidatas do sistema de controle de mercearia:

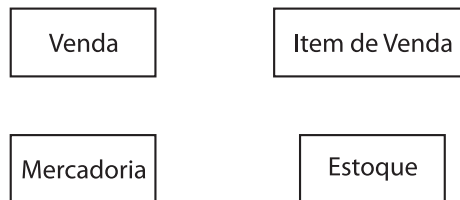



Figura 6.7 - Classes Candidatas

Apesar da possibilidade de utilizar estas técnicas, o grande universo de informações do domínio do problema muitas vezes dificulta a identificação. Existem algumas dicas que podem ser utilizadas para descobrir as classes de um domínio de aplicação observando os seguintes elementos:

- informações que devem ser armazenadas, transformadas, analisadas ou manipuladas de alguma forma;
- outros sistemas e **terminadores externos** que se comunicam ou interagem com o sistema em questão, atentando para as informações que estão sendo trocadas;
- dispositivos físicos com os quais o sistema em estudo terá de interagir, sem considerar a tecnologia para implementar o sistema em si;
- modelos, bibliotecas de classe e componentes gerados em projetos anteriores.

COAD (1992) também sugere alguns indicadores:

Mas, atenção: uma classe que possui uma única instância não deve ser considerada uma classe.

- coisas que são parte do domínio de informação do problema;
- ocorrências ou eventos que precisam ser registrados e lembrados pelo sistema;
- papéis desempenhados pelas diferentes pessoas que interagem direta ou indiretamente com o sistema;
- locais físicos ou geográficos e lugares que estabelecem o contexto do problema;

- unidades organizacionais (departamentos, divisões, etc...) que possam ser relevantes para o sistema.

Você lembra do projeto da clínica Bem-Estar visto na Unidade 2 - 3ª atividade de auto-avaliação? Que tal definir as possíveis classes candidatas?

- Classe Paciente
- Classe Médico
- Classe Convênio
- Classe Laboratório
- Classe Agenda
- Classe Ficha_Médica

Agora quais os atributos que você considera pertinentes para a Classe Convênio?

Você pode listar para a Classe Convênio atributos como:

- Código_Convênio
- Nome_Convênio
- Telefone_Convênio
- Características_Convênio
- Status_Convênio

Se você listar os possíveis atributos da Classe Médico:

- Nome_Médico
- CRM_Médico
- Endereço_Médico
- Telefone_Médico
- Celular_Médico
- Especialidades_Médico
- Horário_Médico

SEÇÃO 3 - Como ocorrem os relacionamentos entre objetos?

Um relacionamento representa a interação entre as classes e objetos, ele apóia o refinamento das classes.

Existem diferentes tipos de relacionamentos possíveis entre as classes identificadas. Os mais importantes são as associações, as dependências e as generalizações.

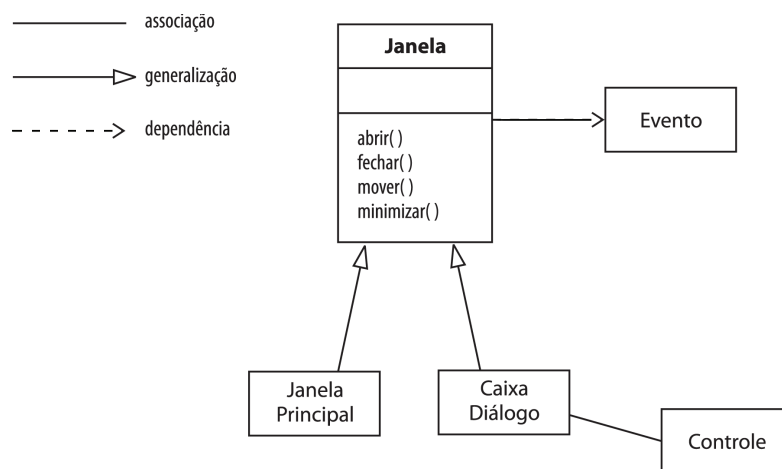



Figura 6.8 - Relacionamento entre objetos.
Fonte: Adaptação Booch (2000).

Conheça então, a partir de agora, as principais características sobre os tipos mais comuns de relacionamentos entre objetos.

a) Relacionamento de Associação

Segundo Furlan (1992), associação é uma relação que descreve um conjunto de vínculos entre elementos de modelo: quando duas classes ou mesmo uma classe, consigo própria, apresenta interdependência; ou quando uma determinada instância de uma das classes origina ou se associa a uma ou mais instâncias da outra classe, você pode dizer que se tem um relacionamento de associação.

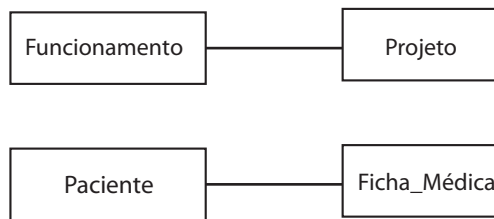



Figura 6.9 - Relacionamento de Associação entre as Classes

Quando você fala em associações é possível representar a quantidade de objetos aos quais o outro objeto está associado. Um exemplo prático: um projeto existe sem que seja alocado um funcionário para este projeto? Quantos projetos podem ser alocados para cada funcionário? Quantos funcionários podem ser alocados para cada projeto?

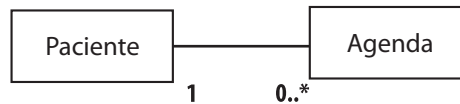
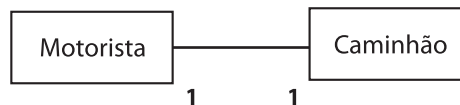
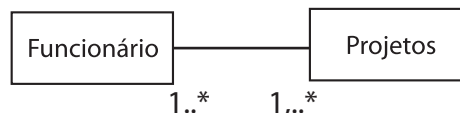
Na notação UML isto é chamado de **multiplicidade**. Alguns autores também utilizam o termo **cardinalidade** (já estudada na Unidade 3). Mas, como representá-lo em um diagrama?

Observe então, a tabela a seguir que apresenta esta simbologia:

Tabela 6.3 - Simbologia UML

Nome	Simbologia
Apenas um	1
Zero ou muitos	0..*
Um ou muitos	1..*
Zero ou um	0..1
Intervalo específico	$1_i..1_s$

Veja alguns exemplos de multiplicidade:

- Um paciente possui nenhumou vários agendamentos.

- Em uma empresa de transporte um motorista dirige apenas um caminhão, e cada caminhão pode ser dirigido por apenas um motorista.

- No terceiro exemplo um funcionário deve estar locado a um ou mais projetos. E cada projeto tem pelo menos 1 funcionário alocado.

Imagine um projeto para uma mercearia, onde se pretende controlar os fornecedores, seus produtos e os pedidos de compra de produtos.

Imediatamente você identifica pelo menos 3 classes candidatas:

- Classe Fornecedor (que vai armazenar dados como endereço, nome, telefone...)
- Classe Produtos (que deve armazenar dados como código, descricao, unidade, ...)
- Classe Item de Compra (que deve armazenar preço, quantidade comprada...)

Você tem a classe fornecedor que pode ter de 0 a n produtos (ou seja o fornecedor oferece à mercearia vários produtos para compra; veja o exemplo de uma revenda de bebidas que possui diferentes tipos de refrigerante e cerveja), os produtos por sua vez podem ter de 0 a n fornecedores (posso ter mais de um fornecedor para o mesmo refrigerante).

Mas cada item de compra pode ter apenas um fornecedor, mas cada fornecedor pode ter de 0 a n itens de compra (no momento da compra vou ter um fornecedor apenas para aquele determinado item para aquela determinada nota fiscal).

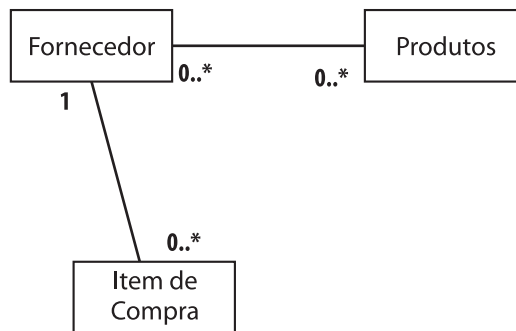



Figura 6.10 - Multiplicidade (Adaptação Pádua, 2000).

Há várias maneiras de nomear associações. No entanto, você deve escolher o nome pensando na descrição da natureza do relacionamento.

Prefira o uso de um verbo ou uma frase verbal. Além de indicar um nome você pode ainda indicar a direção da leitura da associação inserindo um triângulo de orientação.

A figura a seguir mostra a classe “Paciente solicita Agenda” e a indicação da direção que você deve ter a associação

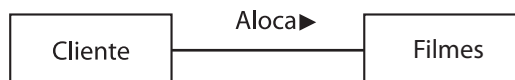



Figura 6.11 - Nomeando uma Associação

Além destes recursos, observe que a classe, ao participar de uma associação recebe um papel específico que é utilizado em um dos lados de uma associação com a finalidade de indicar qual o papel que a classe a seu lado apresenta para a classe do lado oposto.

Um papel define o propósito ou capacidade de uma classe, utilize substantivos para indicar os papéis.

Na figura 6.12, você tem a classe Pessoa desempenhando o papel de funcionário na associação com a classe Banco, que desempenha o papel de empregador.

Figura 6.12 - Papéis em uma Associação

Para essa associação, lê-se: "Pessoa trabalha no banco".

O que significa agregação para o relacionamento de associação?

A agregação é um caso particular da associação. A agregação indica que uma das classes do relacionamento é uma parte ou está contida em outra classe. Mas, as duas classes estão no mesmo nível, ou seja, não existe uma classe mais importante do que a outra na associação.

As palavras-chave usadas para identificar uma agregação são: **consiste em**, **contém** ou **é parte de**. Outra dica importante é que as partes não morrem obrigatoriamente com o todo, e uma mesma parte pode estar em mais de um **todo**. Graficamente você vai representar a associação de agregação por uma linha e um diamante aberto na extremidade.

Observe o exemplo de uma transportadora, onde pode-se dizer que, a empresa tem departamento; a transportadora contém caminhões.

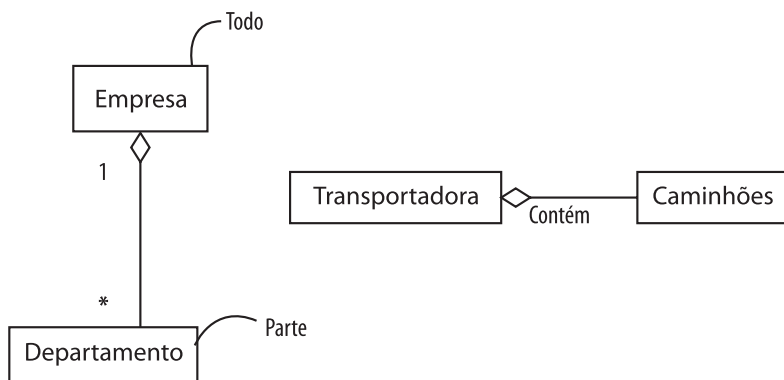



Figura 6.13 - Agregação (BOOCH, 2000).

A **composição** é um tipo especial de agregação onde a multiplicidade do lado "todo" é sempre 1. As partes vivem e morrem obrigatoriamente com o todo, uma mesma parte não pode estar em mais de um "todo". Os objetos da classe parte não existem de forma independente da classe todo.

A composição, segundo RUMBAUGH (1994), é um tipo forte de associação, onde um objeto agregado é composto de vários objetos componentes.

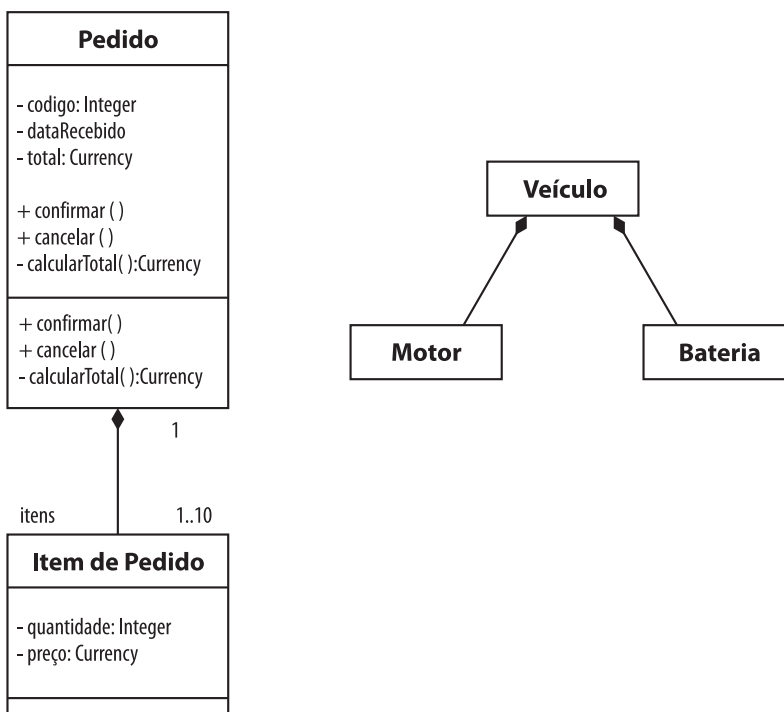



Figura 6.14 - Composição

Na representação das classes, de acordo com a figura 6.14, a associação de composição exprime que o “item de pedido” não existe sem o “Pedido”, ou seja , o “item de pedido” não existe de forma independente no sistema.

A classe “motor” e “bateria” neste exemplo não existem de forma independente da classe veículo. Elas fazem parte do todo “Veículo”.

Na figura 6.15 você vê um pequeno exemplo de classes para um sistema de controle de turmas em uma universidade. Neste exemplo a universidade oferece cursos aos alunos, os cursos por sua vez oferecem disciplinas ministradas por professores aos alunos matriculados.

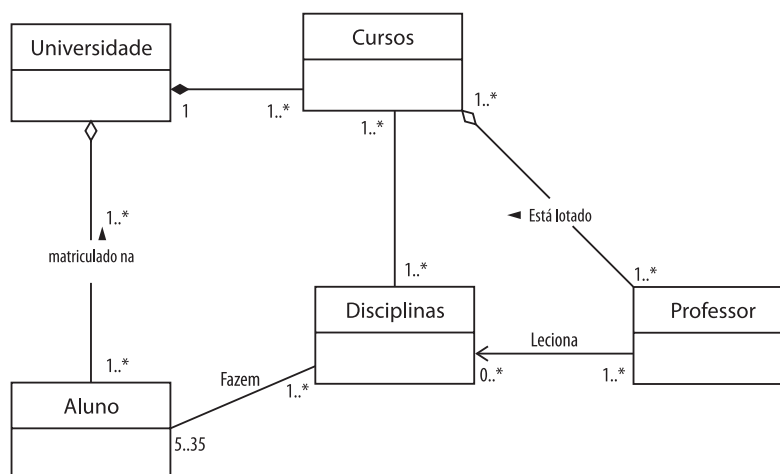



Figura 6.15 - Relacionamentos entre classes

As classes Universidade e Cursos possuem um relacionamento de composição, ou seja a Classe Curso não existe sem a existência da Classe Universidade, se a Classe Universidade for extinta automaticamente a classe Cursos deixa de existir.

Já as classes Aluno e Professor apresentam um relacionamento de agregação pois fazem parte de outra classe. A classe Aluno está contida na classe Universidade e a classe Cursos contém a classe Professor.

As classes associativas estão ligadas a associações e não a outras classes. Simplificando existem situações na análise onde atributos e operações são partes do relacionamento como um todo e não de cada uma das classes envolvidas, neste caso ao invés de se associar ,estes atributos/operações, a um participante é criada uma classe associativa que absorve estes atributos/operações.

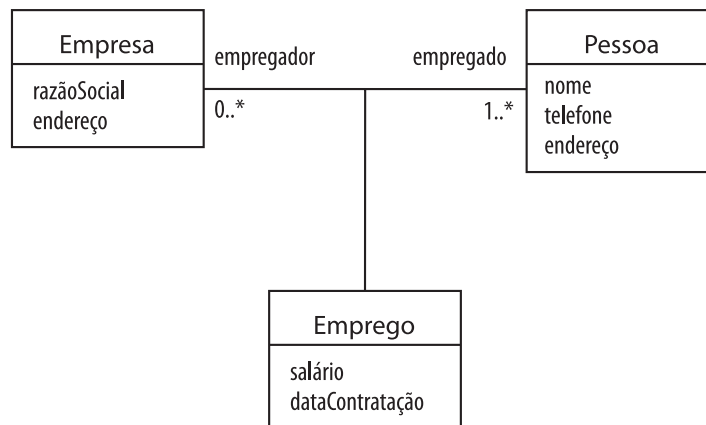



Figura 6.16 - Classe associativa (PÁDUA, 2000).

No exemplo ilustrado na figura 6.16 você vê uma situação onde uma pessoa trabalha em várias empresas, e uma empresa tem vários empregados. Os atributos salário e dataContrataçao não pertencem a classe Empresa nem a classe Pessoa (que mantém dados cadastrais do empregado). Neste caso uma classe associativa Emprego foi criada para comportar estes atributos para cada par (empregado/empregador).

b) Relacionamento de dependência

A dependência indica a ocorrência de um relacionamento entre dois ou mais elementos onde uma classe cliente é dependente de algum serviço da classe fornecedora.

Quando você precisar indicar que um item depende de outro utilize os relacionamento de dependência.

Bezerra (2000) indica situações que levam a um relacionamento de dependência, como:

- **dependência por atributo** – onde a classe A possui um atributo cujo tipo é B;
- **dependência por variável global** – a classe A utiliza uma variável global cujo tipo é B;
- **dependência por variável local** – A possui alguma operação cuja implementação utiliza uma variável local do tipo B;
- **dependência por parâmetro** – A possui pelo menos uma operação que possui pelo menos um parâmetro cujo tipo é B.

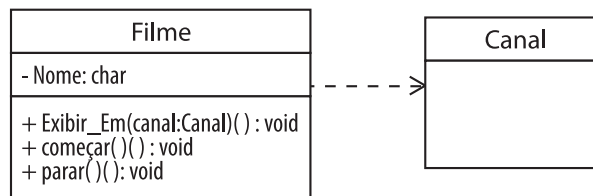



Figura 6.17 - Dependência

Para que as operações da classe Filme sejam executadas a existência da classe Canal é fundamental, pois existe uma dependência de parâmetros entre as classes.

c) Relacionamento de generalização

A generalização é um relacionamento entre itens gerais (super classes) e tipos mais específicos desses itens (as subclasses). **A subclasse herda as propriedades da super classe, principalmente atributos e operações.**

Um relacionamento de especialização/generalização indica que objetos do elemento especializado (subclasse) podem substituir os objetos do elemento generalizado (superclasse). A subclasse tem todos os atributos e operações da superclasse, porém pode ter outros atributos e operações.

Imagine a seguinte situação: você tem uma classe em seu sistema chamada Funcionários. Esta classe possui atributos como nome, endereço, entre outras informações. Existe, no entanto, um tipo de funcionário chamado motorista.

Este funcionário possui atributos específicos como itinerário, horário das rotas. Os motoristas fazem parte da classe funcionários, mas por suas características específicas formam outra classe. Neste caso, a classe motorista herda as características da classe funcionário.

A generalização também conhecida como herança pode ser simples ou múltipla.

- **Herança simples**: a subclasse herda estrutura e ou comportamento de uma única superclasse.
- **Herança múltipla:** a subclasse herda a estrutura e o comportamento de mais de uma superclasse.

Mas, talvez você esteja se questionando: o que uma subclasse pode herdar de uma superclasse?

A subclasse pode herdar atributos, operações e relacionamentos. Além das características herdadas a subclasse possui atributos, operações ou relacionamentos adicionais.

Na figura 6.18, a classe Imóvel possui atributos e operações comuns como área, endereço, IPTU. Mas Apartamento possui características específicas como valor do condomínio, fundo de reserva. O mesmo acontece com a classe Casa. Assim as sub-classes possuem todas as características da super-classe, e além disto ainda possuem características especificas de cada sub-classe. As sub-classe são especializações da super-classe Imóvel.

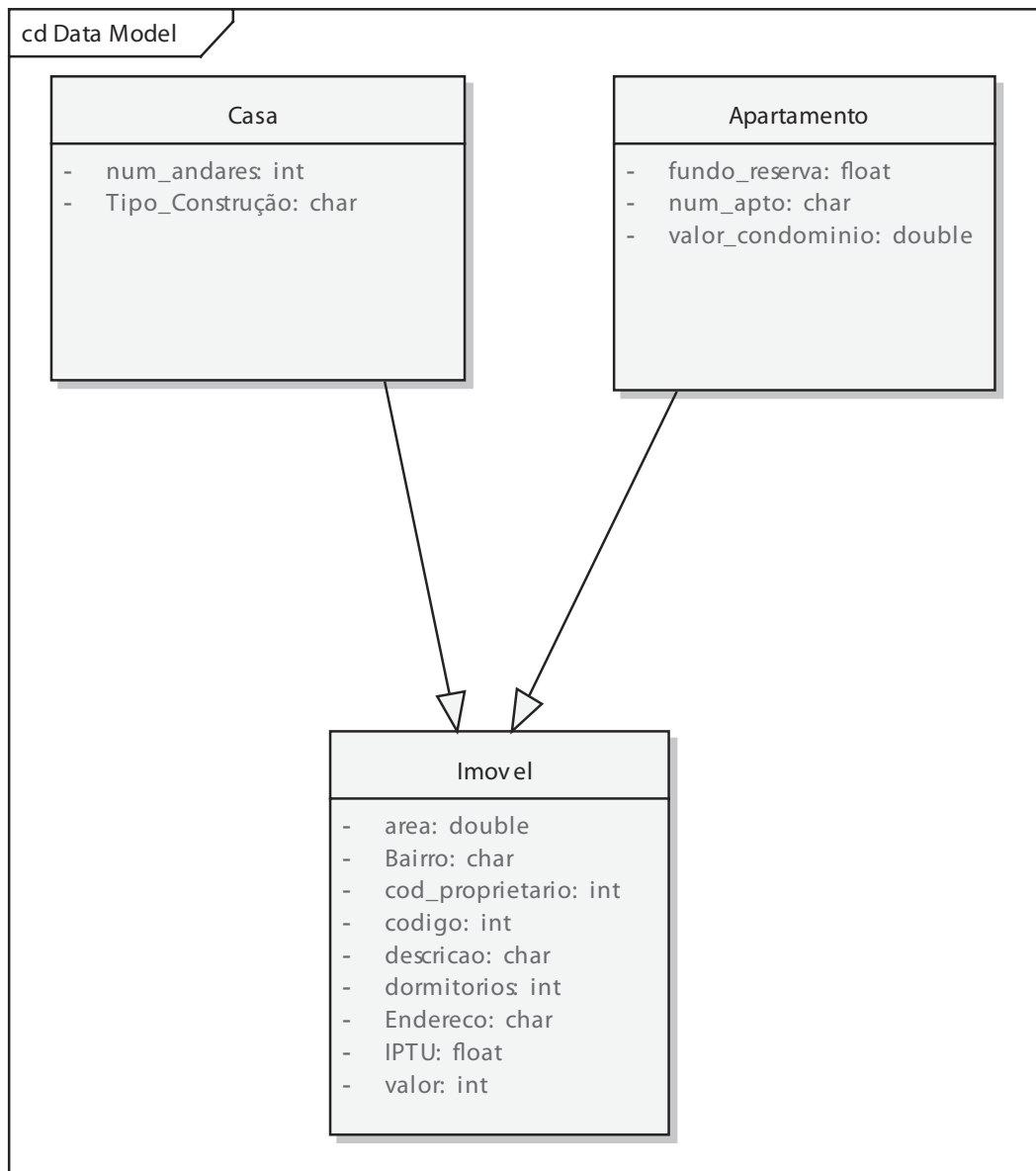



Figura 6.18. Relacionamento de Herança

Agora relembrando o Sistema da Vídeo Locadora observe os relacionamentos entre as classes candidatas propostas:

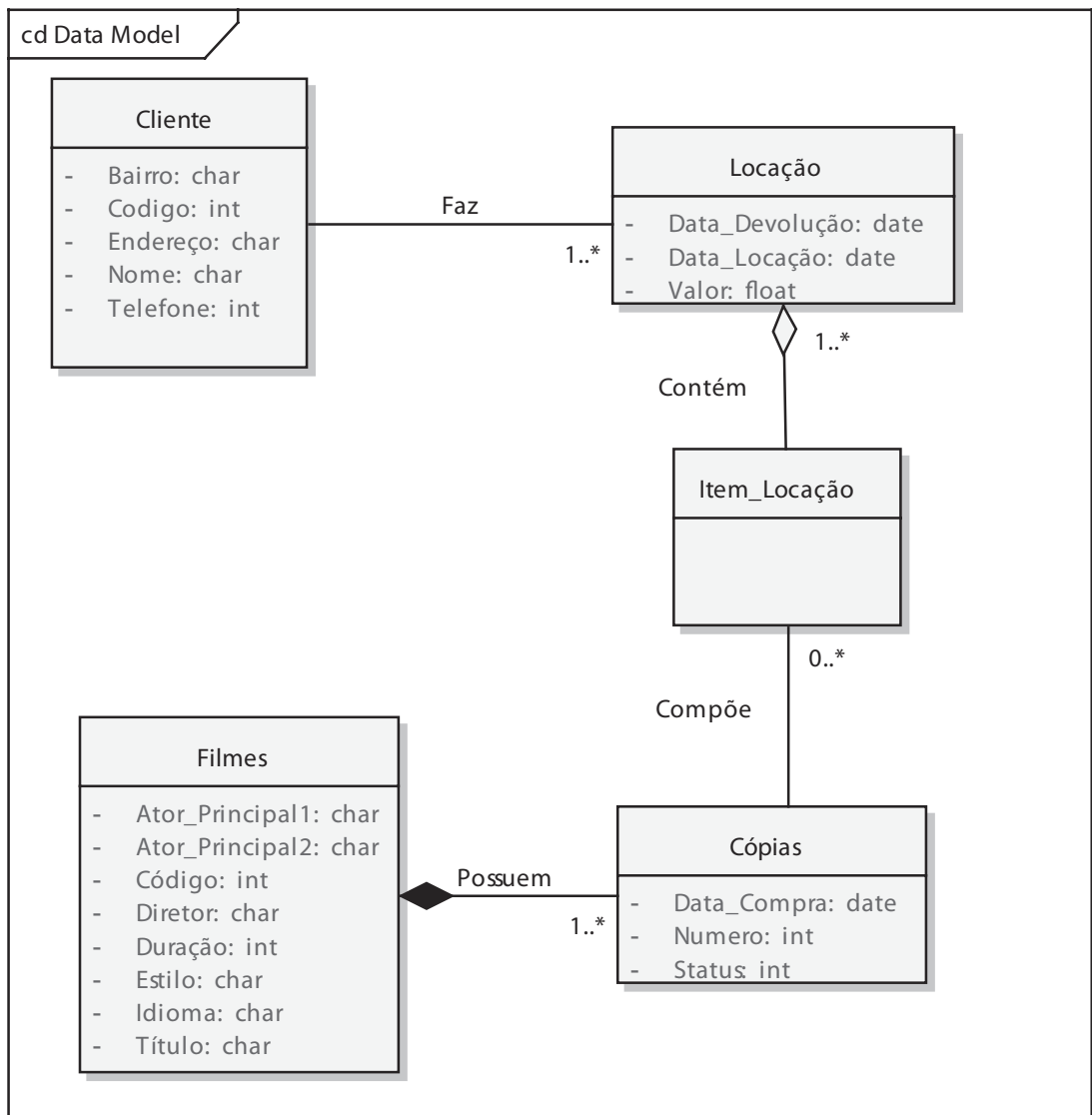



Figura 6.19. Diagrama de Classes Sistema Vídeo Locadora

Você lembra do exemplo sobre o sistema bancário?

Imagine uma situação onde você é convidado a desenvolver um projeto para um caixa eletrônico bancário. O projeto prevê o atendimento dos seguintes requisitos funcionais:

- O sistema deve permitir ao cliente a emissão de saldo somente da conta corrente
- O sistema deve permitir ao cliente a emissão de extrato somente da conta corrente
- O sistema deve permitir a atualização dos dados cadastrais do cliente
- O sistema deve permitir o saque em dinheiro no caixa eletrônico
- O sistema deve permitir a consulta a toda a movimentação financeira do cliente (conta corrente, poupança e aplicações) no caixa eletrônico
- O acesso as funcionalidades do sistema deve ser possível somente após a verificação da conta e senha do cliente ou gerente.

Observe o possível diagrama de classes para este exemplo:

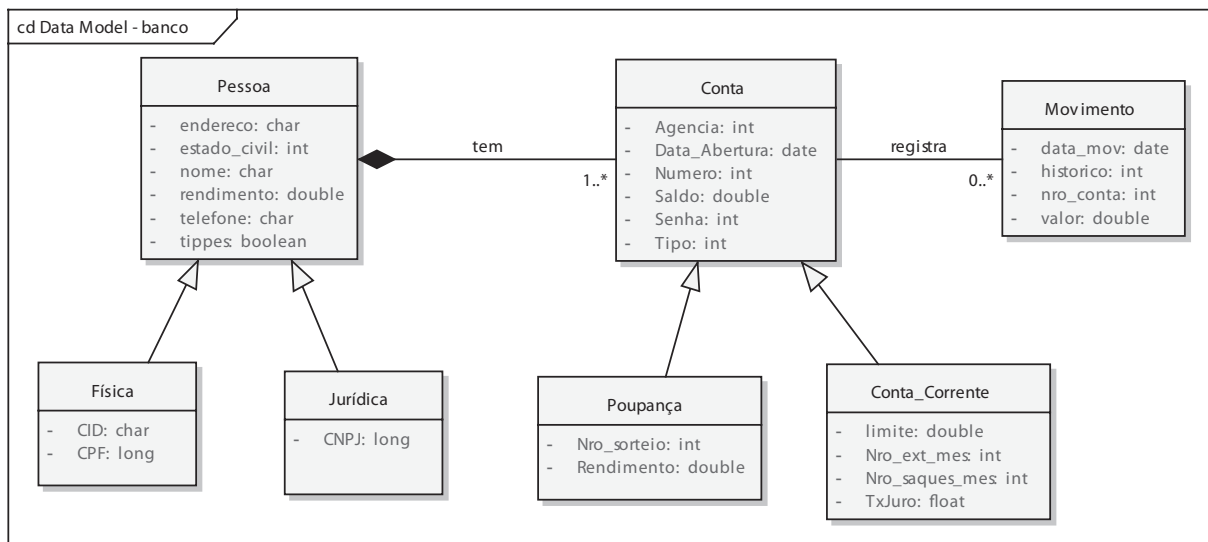



Figura 6.20. Diagrama de Classes Sistema Caixa Eletrônico

SEÇÃO 4 - Como ocorre a divisão das classes do Modelo de Análise?

A divisão ou categorização das classes está fortemente relacionada com as futuras mudanças no sistema.

Com a divisão das classes a partir de suas responsabilidades, no momento em que forem necessárias alterações no sistema, estas passam a ser pontuais.

As classes do modelo de análise podem ser divididas em três camadas:

a) Classes de fronteira;

b) Classes de Entidade;

c) Classes de Controle.

Conheça a partir de agora, as características de cada uma dessas camadas de classes.

a) Classes de fronteira

As classes de fronteira tratam da comunicação com o ambiente do produto, modelando as interfaces do produto com usuários e outros sistemas (entrada e saída de dados).

Cada formulário usado pelo programa é um objeto criado por uma classe de fronteira. A classe de fronteira faz a interface entre um ator e o sistema, uma para cada formulário, relatório ou interface com outro sistema.

Segundo Bezerra (2000), as classes de fronteira tem tipicamente as seguintes responsabilidades:

- notificar as classes de controle sobre eventos gerados externamente ao sistema;
- notificar atores do resultado da interação entre os objetos internos.

São alguns exemplos de classes de fronteira: Formulário Cadastro Cliente, Formulário Cadastro Fitas e Formulário Movimentação Fitas.

b) Classes de entidade

Modelam informações persistentes do sistema, normalmente independentes da aplicação ou entidades do mundo real.

As classes de entidade criam objetos que gerenciam dados. Assim, você pode vê-los de forma correspondente ao banco de dados. Um ator não tem acesso a uma classe entidade e a comunicação ocorre por meio de outros objetos.

Exemplos de classes de entidade: Cliente, Filmes, Locação, Cópias.

As classes de entidades armazenam as informações mantidas pelo sistema. Também é importante para o projeto uma definição da performance esperada no acesso aos objetos da classe.

Você lembra do estudo de caso apresentado no artigo "A importância de utilizar UML para modelar sistemas: estudo de caso", estudado na Unidade 5? Ele se encontra na midiateca. Este estudo discorre sobre um sistema de vendas de CDs musicais pela internet. A figura 6.21 mostra as classes persistentes encontradas para este projeto.

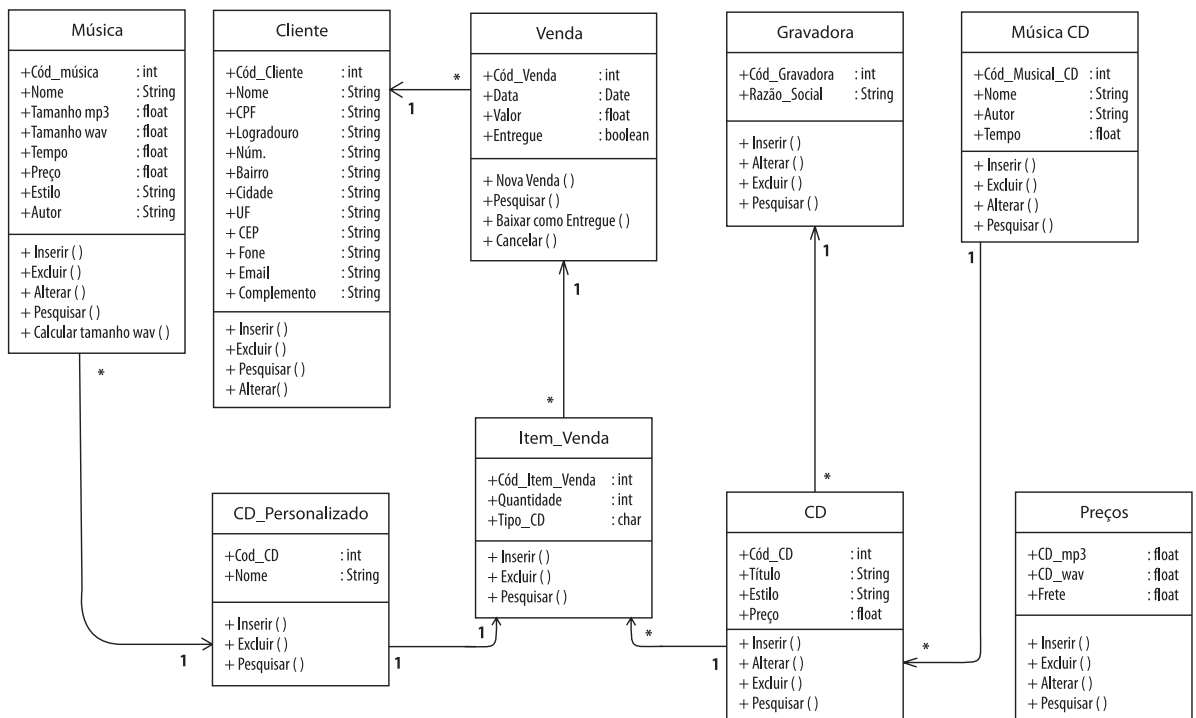



Figura 6.21 - Diagrama de Classes Persistente
Fonte: Figueira, (2005).

c) Classes de controle

Objetos de controle são pontes de comunicação entre objetos de fronteira e objetos de entidade.

Essas **classes** são responsáveis por controlar a lógica de execução correspondente ao caso de uso. Você pode dizer que elas representam a lógica do caso de uso, requisitam e consultam informações das classes de entidade e de interface, bem como não gerenciam dados nem têm saída visível.

As classes de controle atuam entre as classes de interface e as de negócio, isto é, uma para cada caso de uso.

Bezerra (2000) define algumas responsabilidades para as classes de controle:

- realizar monitorações respondendo eventos gerados pelas classes de fronteira;
- coordenar a realização do caso de uso por meio de mensagens das classes de entidade e de fronteira;

- assegurar que as regras de negócio do caso de uso estão sendo seguidas;
- coordenar a criação de associações entre classes de entidade.

São exemplos de classes de controle: Controlador Cliente, Controlador Entrada Fitas, Controlador Saída Fitas, Controlador atrasos.

Lembre-se que os casos de uso complexo devem ser escritos em classes de controle.

O uso da classe de controle é opcional em um sistema. Você deve avaliar claramente, o objetivo da classe de controle é comportar a lógica e as regras de negócio complexas. Isto significa que ações como inclusão, alteração, consultas de cadastros podem tranquilamente ser implementadas em uma classe de fronteira.

Você lembra do *software* de declaração do imposto de renda disponibilizado pelo Governo Federal? As interfaces do sistema com o usuário podem ser descritas pelas classes de fronteira (tela de cadastro, de registro de bens, entre outras), os dados existentes sobre o trabalhador, rendas, despesas e bens são descritos pelas classes de entidade e o cálculo do imposto de renda será descrito na classe de controle.

A representação das Classes na UML se dá pela seguinte notação:

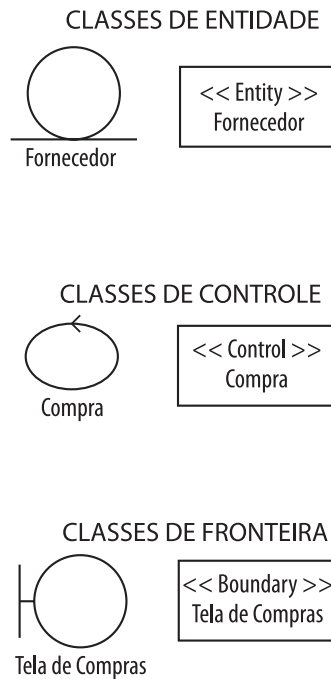



Figura 6.22 - Estereótipos da classe

Você deve estar se perguntando porque uma classe de entidade não deve cuidar de aspectos relacionados as entradas e saídas ? Porque é sugerido o uso de uma classe de fronteira?

Imagine no sistema de vídeo locadora, a aplicação foi desenvolvida para Desktop, mas agora o cliente deseja que o software rode na Internet (o que significa uma interface bastante diferente). Bem, se o projeto foi constituído considerando as três classes de domínio. A equipe de desenvolvimento deve apenas reconstruir as classes de fronteira onde temos as telas do sistema, isto contribui para a eficiência da manutenção do produto.

Para o sistema de vídeo locadora você pode ter algumas classes candidatas:

- Classes de entidade: Filme, Cliente, Locação, Cópias
- Classes de fronteira: A interface para o cadastro do Cliente (Form_Cliente), interface para o cadastro do Filme (Form_Filme), interface para o o movimento da locação (Form_Locação) são alguns exemplos.

- Classes de controle: O cadastro do cliente pode ter uma classe de controle para implementação de métodos (Ctrl _Cliente) assim como o movimento da locação (Ctrl_ Locação)

O que fazer então, depois de realizada a etapa de identificação das classes?

Finalizada a etapa de identificação das classes de controle, fronteira e entidades você pode organizar estas classes em **pacotes lógicos**.

Pacotes lógicos são agrupamentos de elementos de um modelo. O uso de pacotes agrupa classes que possuem um critério comum, facilitando a comunicação.

Quando uma coleção de classes colaboram entre si para realizar um conjunto coeso de responsabilidades, elas podem ser vistas como um subsistema.

De acordo com PRESSMAN (2000), quando visto de fora, um subsistema pode ser tratado como uma caixa preta que contém um conjunto de responsabilidades e suas próprias colaborações.

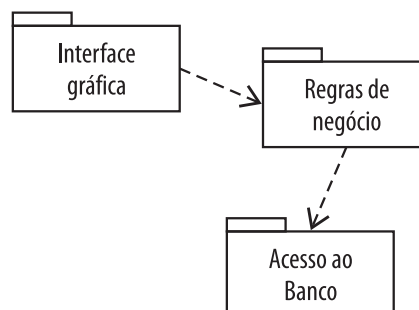



Figura 6.23 - Pacotes lógicos

É importante ressaltar que o pacote deve ter um nome único e textual.

SEÇÃO 5 - O que é diagrama de objetos?

Os diagramas de objetos são como uma fotografia de um sistema Orientado a Objetos em execução.

O diagrama mostra os objetos, os valores de seus atributos e as ligações entre eles, além de ser estático como o diagrama de classes.

Quando fazer uso deste **diagrama**? Este diagrama pode ser bastante útil quando você estiver modelando uma estrutura de dados complexa.

Em alguns livros você vai encontrar o nome diagrama de instâncias como sinônimo de diagrama de objetos.

O diagrama de objetos é representado por um retângulo com dois compartimentos. Na parte superior você identifica o objeto (sublinhado). Na parte inferior você referencia os atributos com seus valores.

Analise a figura a seguir. Note que na parte superior estão as três classes associadas: Pedido, itemPedido e Produtos. A instância Pedido está associada a duas instâncias do itemPedido que conseqüentemente está ligada a uma instância do Produto.

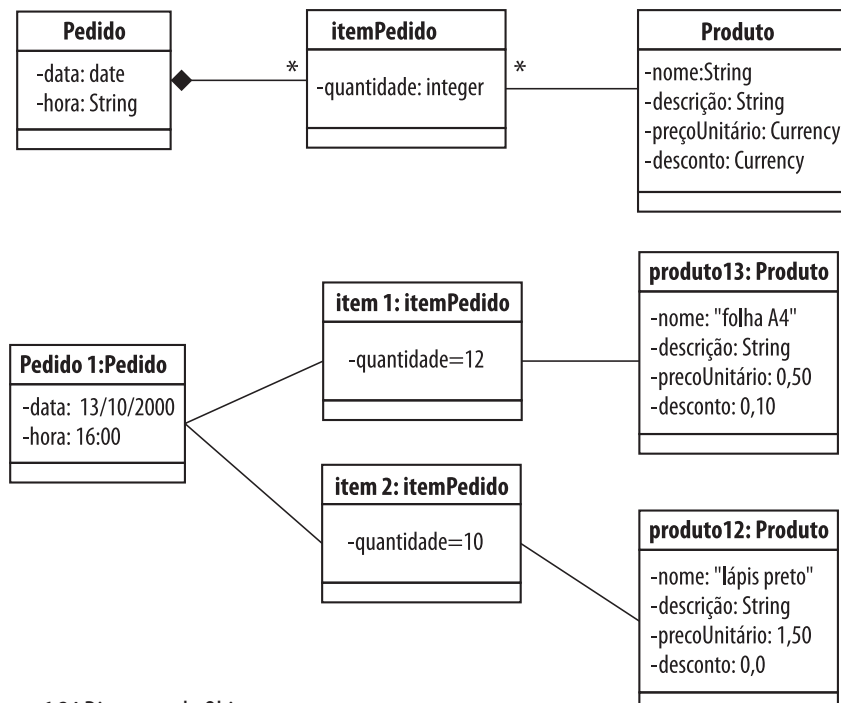



Figura 6.24 Diagrama de Objetos
Fonte: Adaptação de Bezerra (2000).

Mas qual a nomenclatura a ser utilizada na especificação de um diagrama de objetos?

Existem duas possibilidades: os atributos e as operações.

a) Atributos

Os atributos foram apresentados até o momento utilizando-se apenas o nome, mas com certeza em seu projeto você terá que explicitá-los de forma mais detalhada.

A sintaxe a ser apresentada é:

[1]**visibilidade nome:tipo=valor inicial**

A visibilidade se refere ao nível de acesso, o quanto os atributos de um objeto estarão visíveis a outros objetos. Pode ser:

- + Pública – todos têm acesso, podendo ser utilizado por operações declaradas dentro de outras classes.
- # Protegida – pode ser acessado apenas por operações dentro da própria classe, pelas classes da hierarquia e pelas classes do "pacote".
- - Privada – só pode ser acessado por operações dentro da própria classe.

O uso das propriedades de visibilidade apóia um dos conceitos da orientação a objetos: o **encapsulamento**. Assim, você só deixa visível atributos realmente necessários aos demais objetos enquanto que outros atributos tornam-se invisíveis.

Pedido

+ código: int
#data: Data
-total: float
-desconto: float
#dataEntrega:Data

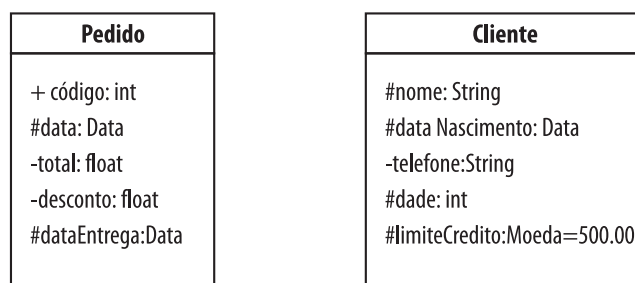



Figura 6.25 - Nomenclatura de Atributos

Sobre a figura 6.25, acompanhe:

- **Nome** – identificador do atributo.
- **Tipo** – o tipo depende da linguagem de programação. Mas, é comum o uso de uma tipologia abstrata onde são definidos os tipos como inteiro, real, caractere, *string*, *float*, *data* ou outra classe.
- **Valor Inicial** – você pode informar um valor inicial para um atributo, quando o objeto da classe for instanciado ele vai pegar o valor automaticamente (veja limiteCredito).

b) Operações

As operações representam o conjunto de funcionalidades da classe. Para cada operação especifica-se sua assinatura:

- **Nome** - identificador para o método.
- **Tipo** - quando o método tem um valor de retorno, o tipo desse valor.
- **Lista de argumentos** - quando o método recebe parâmetros para sua execução, o tipo e um identificador para cada parâmetro.
- ***in*** – parâmetro de entrada, *out* para um parâmetro de saída e *inout* para parâmetros de entrada que podem ser modificados.
- **Visibilidade** - utiliza-se dos mesmos recursos usados para os atributos, definindo o quão visível é uma operação a partir de objetos de outras classes.

Pedido
+obterProduto():String +obterLimite():Moeda +obterMediasComprar():float

Figura 6.26 - Exemplo de operação

Agora que você já estudou as modelagem de classes, aproveite para praticar os conhecimentos conquistados nesta unidade e realize as atividades propostas a seguir.

Atividades de auto-avaliação

Leia com atenção os enunciados e após realize as questões propostas.

1) Assinale a afirmativa correta:

a) ( ) Uma classe é um conjunto de objetos, os objetos por sua vez são identificados por comportamento e estado e nem sempre são únicos.

b) ( ) Uma classe é um conjunto de objetos que compartilham características de atributos, operações, relações e semântica.

c) ( ) É possível dizer que são exemplos de classes em um sistema Universitário: Professor, Aluno, Nome Aluno.

d) ( ) É possível dizer que são exemplos de instâncias de uma classe Professor : "João da Silva", "Ana Luiza".

2) Relacione a primeira coluna com a segunda, indicando a camada mais adequada para cada ocorrência.

A. Classe de Controle	a) ( )	Mensagens de erro para o usuário
B. Classe Persistente	b) ( )	Cálculo da folha de pagamento
C. Classe de Fronteira	c) ( )	Criar ou destruir um objeto (excluir produto)
	d) ( )	Dados do produto
	e) ( )	Telas de consulta de produtos
	f) ( )	Dados do cliente

3) Relacione a primeira coluna com a segunda, indicando as características dos diferentes relacionamentos:

A. Associação

B. Associação Ternária

C. Agregação

D. Herança

E. Dependência

F. Associação Recursiva

G. Classes Associativas

a) ( ) As classes estão ligadas à associações e não a outras classes.

b) ( ) Neste relacionamento ocorre a associação de três classes ao mesmo tempo.

c) ( ) Uma das classes do relacionamento é uma parte da classe, ou está contida em outra classe.

d) ( ) Este relacionamento é possível entre dois ou mais elementos onde uma classe cliente é dependente de algum serviço da classe fornecedora.

e) ( ) Neste caso os objetos da própria classe estão se relacionando.

f) ( ) Quando ocorre este tipo de relacionamento a subclasse herda as propriedades da superclasse, principalmente atributos e operações.

g) ( ) Ocorre quando duas classes ou mesmo uma classe, consigo própria, apresenta interdependência.

4) Na unidade 5, atividade 4, você identificou os casos de uso da clínica "Bem-Estar".

a) Identifique as classes persistentes (nome e descrição da classe)

b) A partir desta identificação construa o diagrama de classes persistentes, apontando os relacionamentos existentes entre as classes.

Síntese

Nesta unidade você aprendeu que um objeto é algo que tem estado, comportamento e identidade. Uma classe é uma definição abstrata de um conjunto de objetos que compartilham uma estrutura e um comportamento comuns, mas que todo sistema engloba muitos objetos que cooperam entre si para produzir a funcionalidade desejada.

A produção das funcionalidades só é possível pela existência de diferentes tipos de relacionamentos como a associação, a dependência e a generalização.

Dentre as características do relacionamento de associação, a multiplicidade é uma das mais importantes. A multiplicidade indica o número de instâncias que participam desta associação. Você também viu que as operações determinam como um objeto age e reage às mensagens que ele recebe e que é possível agrupar estes objetos e classes em pacotes lógicos criando uma visão mais clara sobre o sistema.

Você também estudou a importância de modelar o sistema em diferentes camadas, como as camadas de controle, fronteira e persistente. Esta modelagem cuidadosa facilita futuras manutenções no seu projeto.

Em resumo, esta unidade englobou conceitos e abstrações relacionadas aos aspectos estáticos do sistema. Mas, para o bom andamento do projeto é necessário ter uma visão dinâmica destes objetos, e é exatamente este ponto que será abordado na próxima unidade.

Saiba mais

Para aprofundar as questões abordadas nesta unidade, você poderá pesquisar:

- BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. São Paulo: Campus, 2002. (ler capítulos 5, 6 e 8)
- BOOCH, G. ; RUMBAUGH, J. ; JACOBSON, I. **UML Guia do Usuário**. São Paulo: Ed. Campus, 2000. (ler capítulos 8 e 9).
- PAULA FILHO, Wilson de Pádua. **Engenharia de** ***software*****.** São Paulo: Campus, 2001. (ler capítulo 4).

- Uma boa leitura sobre modelagem de chaves você vai encontrar no capítulo 4 do livro: UML Uma abordagem prática de Gilleanes T. A. Guedes.

Uma boa leitura sobre modelagem de classes você vai encontrar no capítulo 4 do livro: UML uma abordagem prática de Gilleanos T. A. Guedes.

UNIDADE 7

7

Modelos de Interações

Objetivos de aprendizagem

- Reconhecer objetivos e características existentes na modelagem da visão dinâmica do projeto.
- Compreender a notação utilizada nos diagramas que especificam o modelo dinâmico do sistema.
- Entender as características existentes entre as diferentes mensagens utilizadas na comunicação entre objetos.
- Perceber quando o uso do diagrama pode ser interessante em um projeto de *software*.

Seções de estudo

Seção 1 Quais são os elementos do Modelo de Interação?

Seção 2 O que é Diagrama de Interação?

Para início de estudo

Você estudou na unidade 5 sobre o modelo de casos de uso. Este modelo identifica o que o sistema deve fazer e para quem. Apesar de descrever o que e para quem deve ser feito, não são descritos maiores detalhes relacionados ao comportamento interno do sistema na execução das funcionalidades.

O modelo de classes de domínio estudado na unidade 6, agrega ao projeto a visão estática e estrutural do sistema. Sob esta visão, você construiu a definição das classes e responsabilidades. Apesar de você já ter até aqui uma idéia clara do sistema com estes dois modelos, nenhum aspecto relacionado ao mecanismo de comunicação e comportamento entre objetos foi tratado até este momento.

Por isso, nesta unidade, você vai estudar sobre o modelo de interações, que apresenta as mensagens trocadas entre os objetos na execução de um determinado cenário.

O uso do modelo de interações procura descrever o modelo dinâmico do sistema propondo a descrição, inclusive temporal da troca de mensagens entre objetos.

Seção 1 - Quais são os elementos do Modelo de Interações?

Segundo BOOCH (2000), interação é como um comportamento que abrange um conjunto de mensagens trocadas entre um conjunto de objetos em um determinado contexto para a realização de um propósito.

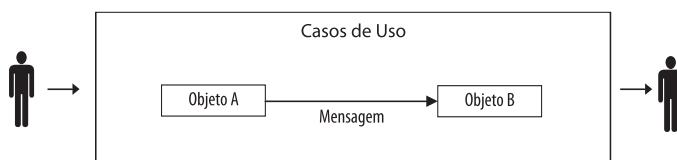



Figura 7.1 - Interação entre objetos

A interação é empregada para a modelagem do fluxo de controle de uma operação, uma classe, um componente, um caso de uso ou do sistema como um todo.

O uso de interações também introduz mensagens que são enviadas de objeto a objeto. Essas mensagens envolvem a chamada a uma operação ou o envio de um sinal.

No decorrer do seu estudo você já leu várias vezes a palavra mensagem, certo? É provável que você já tenha uma idéia sobre o significado desta palavra e por isto nestas três últimas unidades conceituar o substantivo parecia irrelevante. Mas, nesta unidade essa palavra se torna o elemento fundamental do modelo.

Segundo Bezerra (2000), uma mensagem é uma solicitação de execução de uma operação em outro objeto, um objeto pode ainda enviar uma mensagem para si mesmo (mensagem reflexiva).

O uso de uma mensagem em um diagrama de interação permite a passagem de informações que são repassadas para a operação que será executada no objeto receptor.

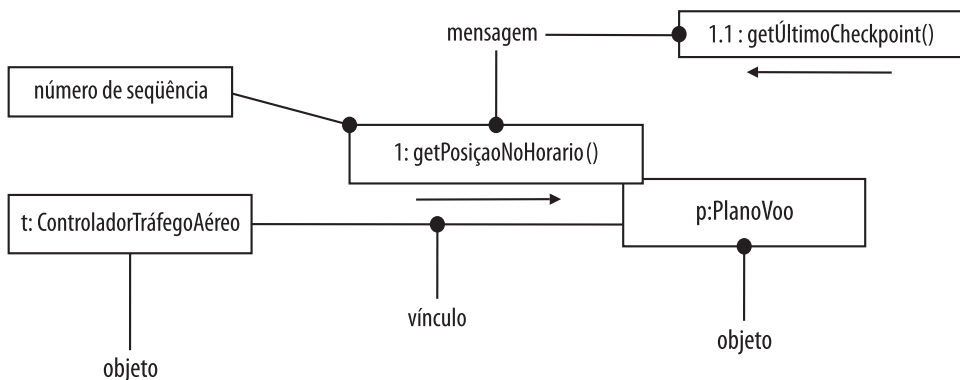



Figura 7.2 - Interação entre objetos de operação de vôo

Na figura 7.2 são considerados objetos:

<u>t:ControladorTráfegoAéreo e p:PlanoVoo</u>

<u>1: getPosiçaoNoHorario()</u> e <u>1.1 : getÚltimoCheckpoint()</u> são consideradas mensagens.

Os números 1 e 1.1 nas mensagens são números de seqüência usados para organizar a sequencialização das mensagens.

As mensagens são representadas graficamente por linhas com uma direção e quase sempre incluem os nomes de suas operações, os objetos e seus vínculos como ilustra a figura 7.2, onde a troca de mensagens ocorre entre os objetos Plano de Vôo e Controlador Tráfego Aéreo.

Os **objetos** de uma interação desempenham determinados papéis como você pode perceber na figura 7.3. Nessa figura, a classe Pessoa representa o papel do Empregado e a classe Empresa, o papel empregador.

Já os **vínculos** constituem normalmente uma instância de uma associação. Um vínculo especifica um caminho pelo qual um objeto pode enviar uma mensagem para outro objeto ou para o mesmo objeto. Observe a figura a seguir:

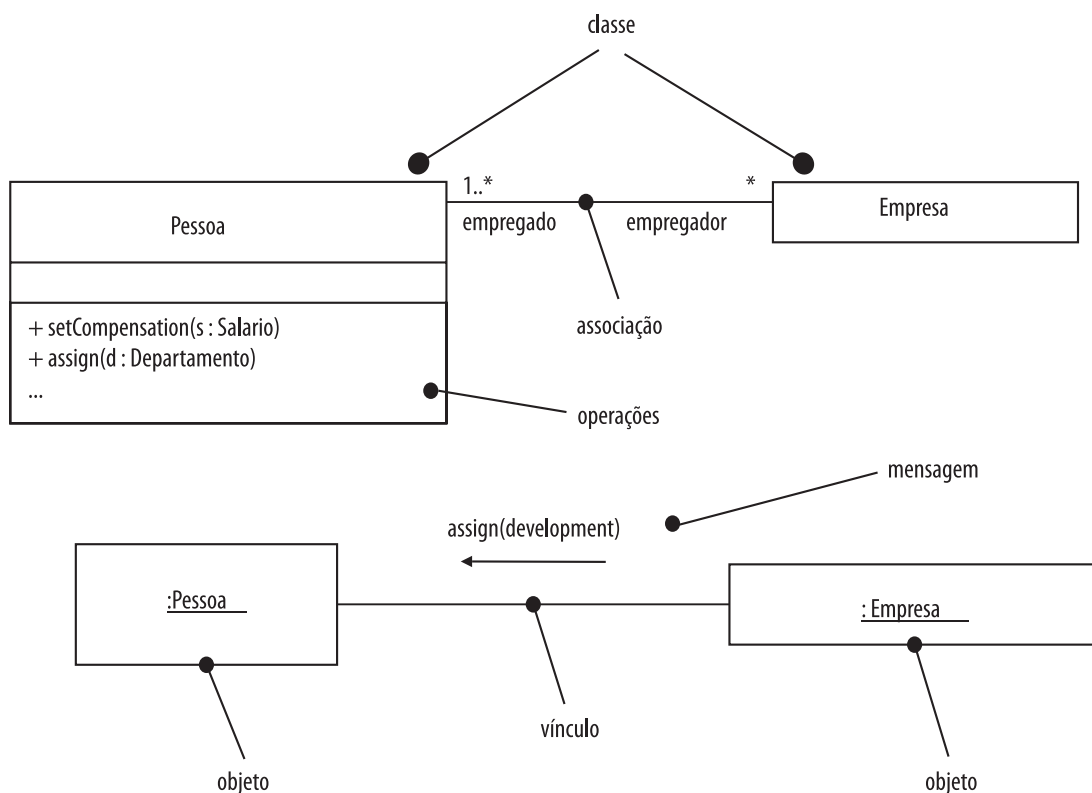
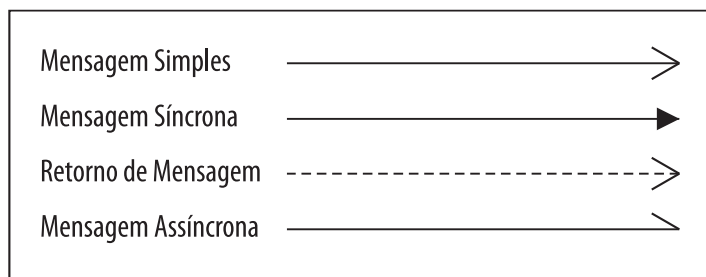



Figura 7.3 - Interação entre objetos
Fonte: Booch (2000).

Quais são os tipos de mensagem?

A notação UML descreve a possibilidade de três tipos de mensagens:

- **Mensagem simples** – este tipo de mensagem é utilizada quando a natureza da mensagem é irrelevante.
- **Mensagem síncrona** – indica que o objeto remetente espera que o objeto receptor processe a mensagem antes de recomeçar o seu processamento. Neste caso, o objeto receptor ficará bloqueado até que a requisição seja atendida.
- **Mensagem assíncrona** – o objeto remetente não espera resposta para prosseguir com seu processamento.

A mensagem é representada por uma seta onde o sentido é do objeto remetente para o objeto receptor. As mensagens possuem um rótulo que procura especificar as informações que a mensagem deve transmitir.

Você pode usar a seguinte sintaxe:

Expressão-seqüência recorrência: v := mensagem

Assim, tem-se:

a) Expressão-seqüência - as mensagens são passadas de um objeto para outro. Este fluxo de mensagens forma uma seqüência. As seqüências devem ter um ponto de partida indicando o início do processo. A expressão de seqüência elimina ambigüidades acerca de quando a mensagem foi enviada em relação às demais.

1: AtenderChamado()

Sequência é o número 1, a expressão AtenderChamado()

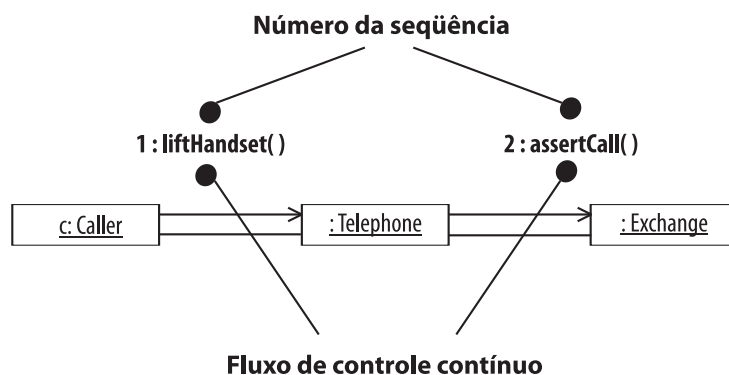



Figura 7.4 Seqüenciamento das mensagens

Fonte: Booch (2000).

Na figura 7.4, observe a numeração das mensagens (1 e 2) que indica a direção e a ordem em que as mesmas acontecem.

b) Recorrência – às vezes o envio de uma mensagem está condicionado ao valor de uma expressão lógica (verdadeiro ou falso) ou ao número de vezes que a mensagem será enviada. Se a recorrência for uma cláusula-condição, então a mensagem será enviada somente se a condição for verdadeira.

Sua sintaxe é sempre entre colchetes: [cláusula condição]

[existe produto estoque] EfetuarVenda()

A repetição é ordenada pelo uso de asterisco: *[cláusula iteração]

*[enquanto Numero_Itens < 10] InserirItem()

c) Variável – identifica uma variável que recebe o valor de retorno da operação executada pelo receptor.

1.2.1: Z :=verificarEstoque(e)

A variável Z vai receber o retorno da operação verificarEstoque.

Quando uma mensagem é enviada você está especificando uma comunicação entre objetos, que possuem uma expectativa de realização de uma atividade. Quando a mensagem é passada o resultado é uma ação na forma de uma instrução executável.

Você pode fazer a modelagem de vários tipos de ação, tais como:

- **Call** – (mensagens síncronas) solicita uma operação em um objeto.
- **Send** – (mensagens assíncronas) envia um sinal para um objeto.
- **Create** – cria um objeto.
- **Destroy** – destrói um objeto.
- **Retorno** – (return) retorna o controle a quem ativou um Call.

Seção 2 - O que é Diagrama de Interação?

Um diagrama de interação mostra uma interação formada por um conjunto de objetos e seus relacionamentos, incluindo as mensagens que poderão ser trocadas entre eles.

O diagrama de interação apresenta o funcionamento interno do sistema para que o ator ao realizar o caso de uso consiga atingir seus objetivos.

Um diagrama de interação pode modelar um caso de uso, assim como pode ser necessário o uso de vários diagramas para modelar a interação de um caso de uso que possui diferentes cenários.

Existem dois diagramas de interação: o diagrama de seqüência e o diagrama de comunicação.

a) Diagrama de Seqüência

Um diagrama de seqüências é um diagrama de interação que enfatiza a ordenação temporal de mensagens, apresentando os objetos que participam da interação e a seqüência de mensagens trocadas.

O diagrama de seqüência descreve o comportamento interno, mostrando os eventos entre objetos, mas omite a associação entre estes objetos.

Para construir um diagrama de seqüência é necessária a prévia definição do diagrama de classes com a indicação das operações associadas.

A descrição é sempre uma interação dentro de uma unidade de tempo. É ideal para a especificação de processos que ocorrem em tempo real.

A notação usada pela UML para representar o diagrama de seqüência utiliza-se de atores, objetos, classes e mensagens, conforme mostra a figura a seguir.

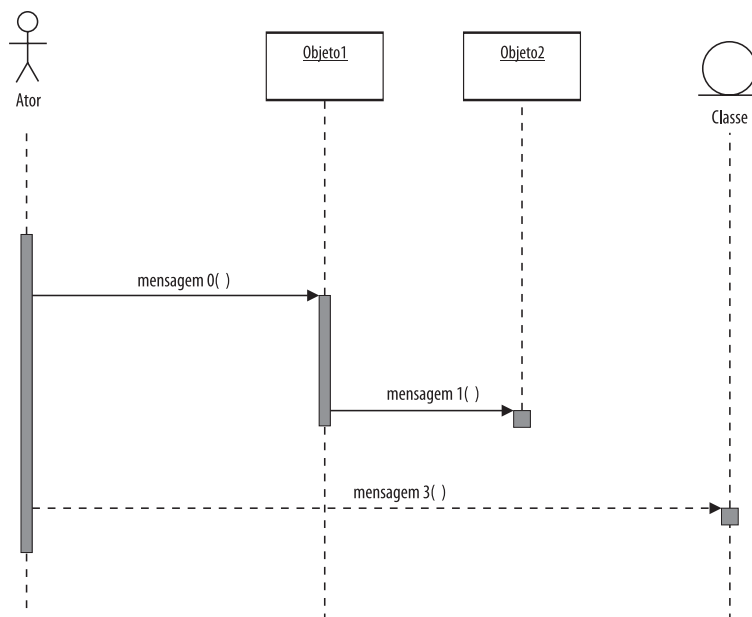



Figura 7.5 - Elementos Gráficos do Diagrama de Seqüência

- **Atores** – participam do diagrama de seqüência opcionalmente, dependendo do cenário do caso de uso.
- **Objetos** – a ordem na qual os objetos aparecem não é pré-definida, mas normalmente utiliza-se a ordem da esquerda para a direita: ator, objetos de fronteira, objetos de controle, objetos de entidade e atores secundários.
- **Classes** - aparece no diagrama quando uma mensagem for endereçada para a classe e não para o objeto.
- **Linhas de vida** – cada objeto aparece no topo de uma linha vertical tracejada, é a linha da vida.
- **Mensagem** – são as linhas horizontais com flechas que ligam uma linha de vida a outra. As flechas horizontais são rotuladas com as mensagens.

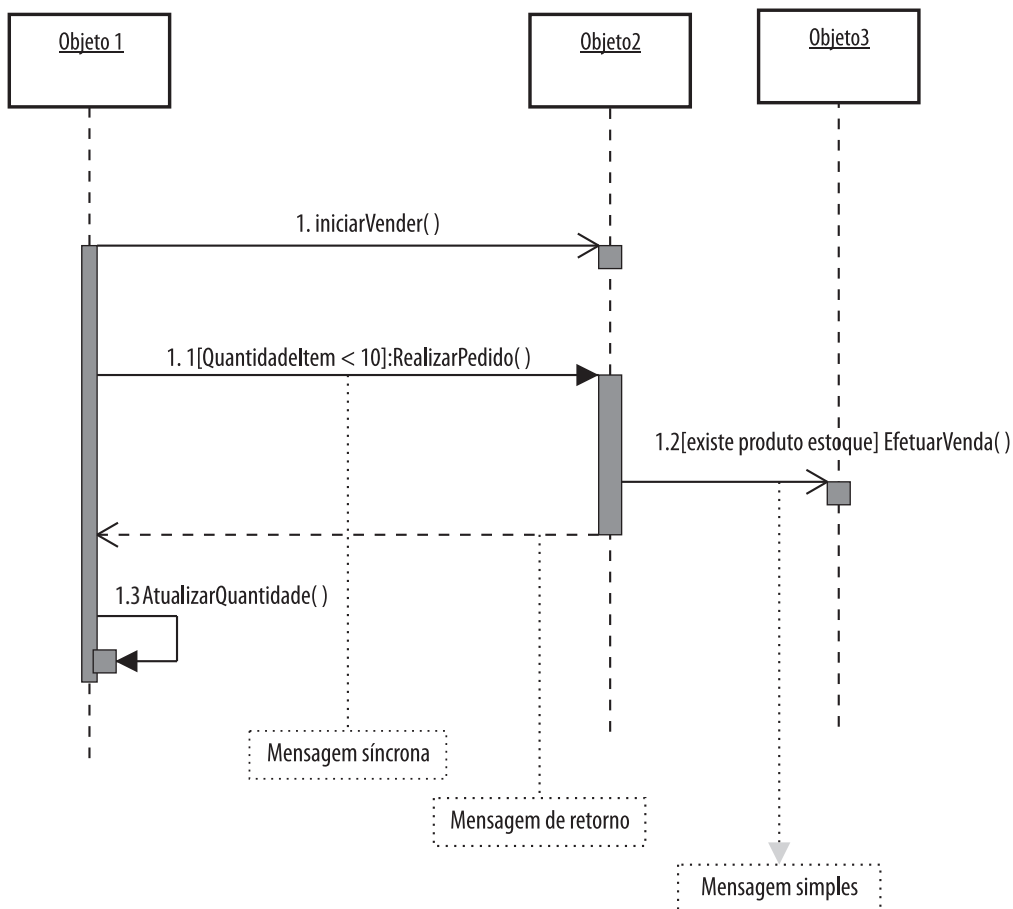



Figura 7.6 - Diagrama de seqüência

- **Focos de Controle** – os focos de controle são as caixas retangulares que estão sobre a linha da vida do objeto. O foco de controle indica o tempo necessário para que o objeto realize uma ação. O início do foco deve estar na altura da flecha de mensagem. O final deve coincidir com o final da atividade realizada pelo objeto.

A figura 7.7 mostra o diagrama de seqüência de um caso de uso para registro de uma venda. O ator Atendente envia uma mensagem para totalizar o objeto Venda. Em seguida o ator dispara a mensagem para registrar o modo de venda para o objeto Venda.

A partir daí se estabelece uma recorrência condicional: se a venda for a prazo envia a mensagem *Inserir*, sendo o parâmetro o próprio objeto Venda para o objeto Contasareceber. Se a venda for a vista, envia a mensagem *registrapagamento* para o objeto Caixa.

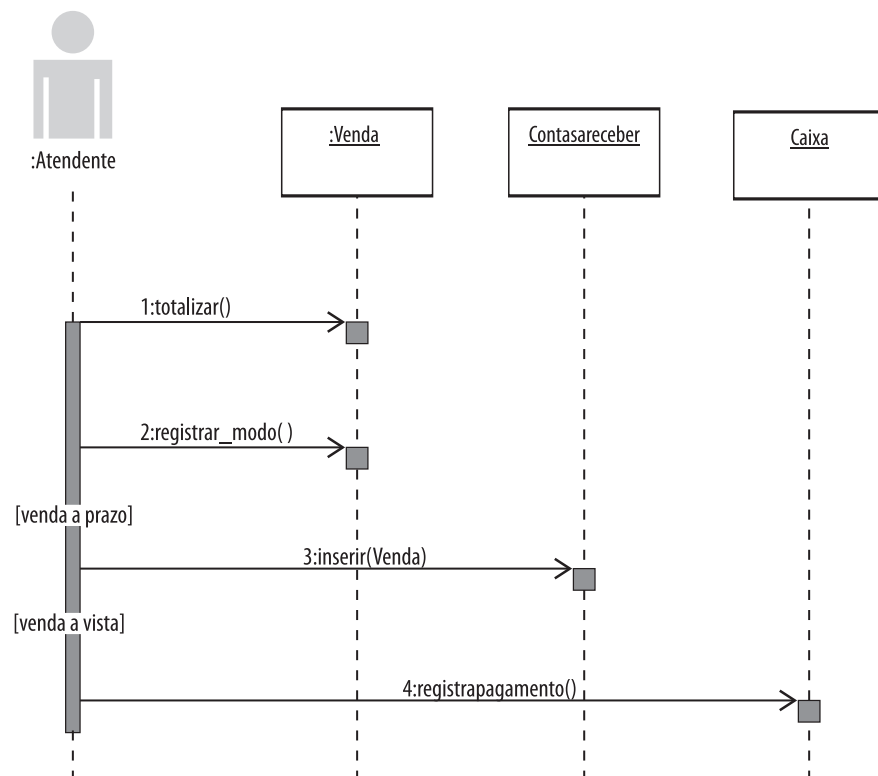



Figura 7.7 - Diagrama de Seqüência

Fonte: Pádua (2000).

Agora observe a figura a seguir:

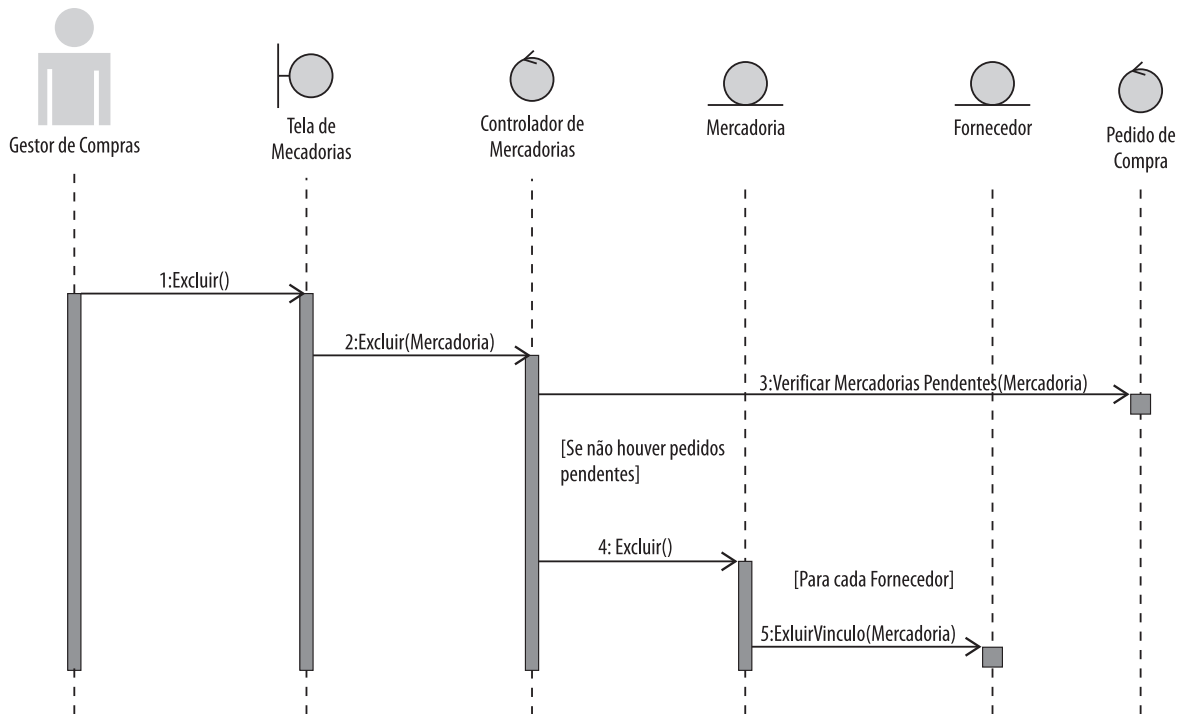



Figura 7.8 - Diagrama de Seqüência

Fonte: Pádua (2000)

No diagrama da figura 7.8 são utilizadas classes de fronteira (Tela de Mercadorias), classes de controle (Controlador de Mercadorias, Pedido de Compra) e classes persistentes (Mercadoria e Fornecedor).

Todas as mensagens estão seqüenciadas (1-5) indicando a ordenação temporal das mensagens.

Não esqueça que o diagrama de seqüência está baseado na descrição do caso de uso, então ele é um reflexo do que foi documentado. Observe o diagrama de seqüência para o sistema de vídeo locadora para o caso de uso Gerenciar Locações. Observe que foram usadas apenas as classes de fronteira e de entidade.

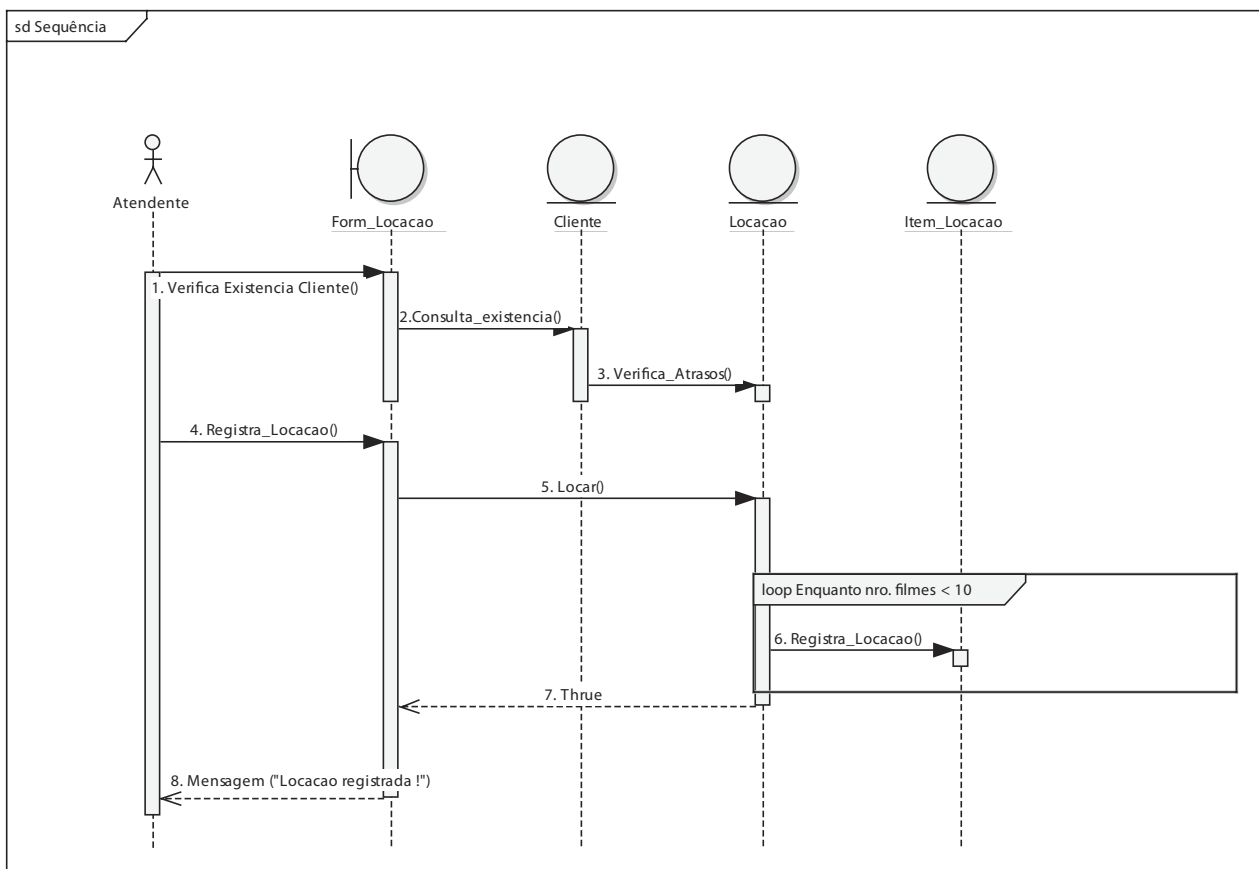



Figura 7.9. Diagrama de seqüência para o sistema de vídeo locadora

b) Diagrama de Comunicação

O diagrama de comunicação é um modo alternativo para representar a troca de mensagens entre um conjunto de objetos. O diagrama de colaboração sempre mostra os objetos relevantes para a execução do caso de uso.

Neste diagrama, você não apresenta a ordem em que as mensagens foram enviadas, pois não existe a dimensão de tempo (linhas da vida do diagrama de seqüência), o que obrigatoriamente tem-se expressões de seqüência em todas as mensagens (1, 1.1, 1.2...).

Diagramas de comunicação apresentam ênfase no sistema, ou seja, são usados para obter uma visão geral do sistema:

- os objetos que são criados durante uma colaboração são especificados como {**new**};
- os que são destruídos durante uma colaboração são especificados com {**destroyed**}
- os que são criados e destruídos na mesma colaboração são especificados com {**transient**}.

A leitura do diagrama ou a seqüência das mensagens é organizada pelas setas junto ao rótulo da mensagem.

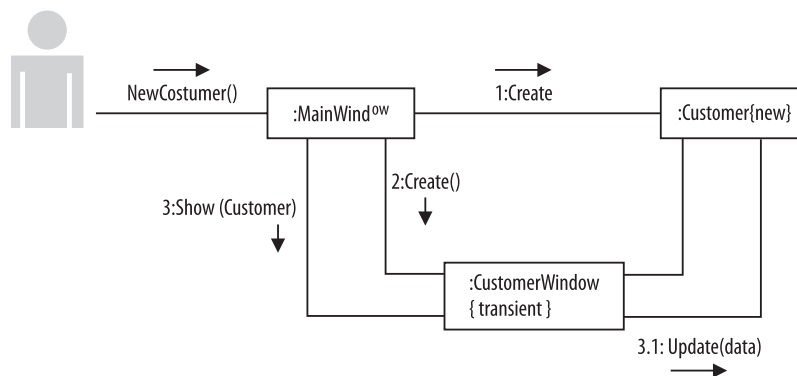



Figura 7.10 - Exemplo de um diagrama
Fonte: Colaboração Ambler (1998).

O objeto MainWindow recebe a mensagem NewCustomer e cria um objeto Customer. Um CustomerWindow é criado e o objeto Customer é então passado para o CustomerWindow, o qual atualiza os dados do Customer.

No exemplo a seguir você vê o diagrama de colaboração de um sistema de empréstimo para uma biblioteca:

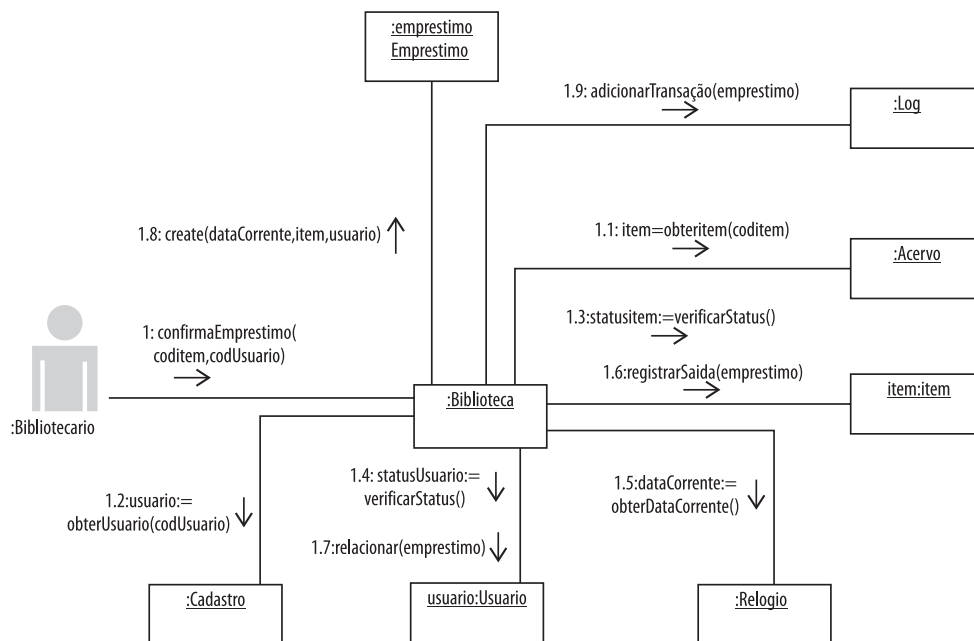



Figura 7.11 - Diagrama de Comunicação
Fonte: Liesenberg (2005)

O diagrama inicia a comunicação pela mensagem confirmarEmpréstimo e se finaliza pela mensagem adicionarTransiçao no objeto Log. Observe que o diagrama apresenta a relação existente entre os diferentes objetos de forma bastante legível pelo uso da numeração e do uso do direcionamento das mensagens por meio das setas.

Afinal, diagrama de seqüência ou diagrama de comunicação?

Um diagrama de seqüência de sistema representa uma sucessão de eventos de entrada, gerados por um ator ao executar um fluxo de um caso de uso.

Nos diagramas de seqüência existe uma linha de vida do objeto. A linha de vida é a linha tracejada vertical que representa a existência de um objeto em um período de tempo.

Além disso, existe o foco de controle que mostra o período durante o qual um objeto está desempenhado uma ação, diretamente ou por meio de um procedimento subordinado.

Este diagrama é interessante na descrição de uma seqüência particular de funcionamento, mas pode ser confuso quando existem muitas seqüências alternativas .

Os diagramas de colaboração sempre apresentam o caminho que indica como um objeto está vinculado a outro, além disto existe o número de seqüência para indicar a ordem temporal de uma mensagem.

Se você precisa de um diagrama que demonstre o fluxo de eventos no decorrer do tempo, então você deve utilizar o diagrama de seqüência, se a ênfase for o contexto do sistema, a melhor opção é o diagrama de comunicação.

Quer conhecer mais?

Para conhecer um pouco mais sobre os modelos de interação, acesse a Midiateca. O texto "Exemplo Seqüencia&Colaboração" apresenta o diagrama de colaboração e de seqüência para um sistema de videolocadora.

Agora para praticar os conhecimentos conquistados nesta unidade, realize a seguir as atividades propostas.

Atividades de auto-avaliação

Leia com atenção os enunciados e após realize as questões propostas:

1) Relacione os conceitos abaixo, observe que uma mesma opção pode se repetir:

A. Cláusula Condição
B. Mensagem
C. New
D. Destroyed
E. Transient
F. Linhas de Vida
G. Cláusula Iteração
H. Call
I. Send

a) ( ) São as linhas horizontais com flechas que ligam uma linha de vida a outra.

b) ( ) Objetos são destruídos durante uma colaboração.

c) ( ) Os objetos aparecem no topo de uma linha vertical tracejada.

d) ( ) 1:[preço < 10,00]:Venda aVista (produto)

e) ( ) Utilizado para mensagens síncronas

f) ( ) Objetos são criados e destruídos durante uma colaboração.

g) ( ) 2:* [lê_codigo]

h) ( ) Utilizado para mensagens assíncronas

i) ( ) Objetos são criados durante uma colaboração

2) Construa o diagrama de seqüência da Clínica Bem-Estar (visto na unidade 5 atividade 4 de auto-avaliação) para o caso de uso Agendar Horário.

3) É correto afirmar que interação:

a) ( ) Propõe a troca de mensagens entre atores.

b) ( ) Mensagens são trocadas entre um conjunto de objetos em um determinado contexto para a realização de um propósito.

c) ( ) A interação pode ser definida como uma solicitação de execução de uma operação em outro objeto.

d) ( ) Na interação, a mensagem síncrona o objeto remetente não esperaresposta para prosseguir com seu processamento, já na assíncrona o objeto remetente espera que o objeto receptor processe a mensagem antes de recomeçar o seu processamento.

Síntese

Nesta unidade você foi apresentado ao modelo dinâmico do projeto. Foi possível perceber que esta visão aborda aspectos internos do sistema, chegando ao nível das funções que serão implementadas futuramente.

O uso de diagramas de interação permite uma visualização e entendimento do funcionamento temporal da troca de mensagens entre os diversos objetos. Você pode representar as mensagens por meio de diferentes notações, onde é possível apresentar relações síncronas, assíncronas e de retorno entre os objetos.

O uso da representação temporal da troca das mensagens pode ou não ser representada. Se a dimensão de tempo é fundamental para o entendimento, você pode utilizar o diagrama de seqüência. Mas se a dimensão do contexto do sistema é o mais importante, utilize o diagrama de colaboração.

Na próxima unidade, você vai iniciar seu estudo sobre o modelo de estados e compreender que os objetos podem modificar seus estados durante o processamento ou permanecer no mesmo estado por toda sua vida útil. A documentação dos estados dos objetos é fundamental, pois suas alterações estão diretamente relacionadas às mudanças no sistema.

Saiba mais

Caso você tenha se interessado em aprofundar seus estudos sobre os assuntos tratados nessa unidade, sugere-se que leia:

- BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML. São Paulo:** Campus, 2002. (capítulo 7).
- BOOCH, G.; RUMBAUGH, J.; JACOBSON, I. **UML Guia do Usuário.** São Paulo: Ed. Campus, 2000. (capítulos 18 e 27).

UNIDADE 8

Modelos de Estados

Objetivos de aprendizagem

- Reconhecer objetivos e características existentes na modelagem da visão dinâmica do projeto.
- Compreender a notação utilizada nos diagramas que complementam a especificação do modelo dinâmico do sistema.
- Reconhecer as possibilidades de utilização e as notações envolvidas no diagrama de estados e no diagrama de atividades.

Seções de estudo

Para início de estudo

O modelo dinâmico do sistema completa-se a partir de cinco diagramas: o diagrama de atividades, o diagrama de seqüência, os diagramas de colaboração, o diagrama de estados e o diagrama de casos de uso. Cada um destes diagramas especifica e esclarece aspectos diferentes do sistema.

Até este momento, você estudou três destes diagramas (casos de uso, comunicação e seqüência). Nesta unidade serão apresentados os diagramas de transição de estado que modelam o comportamento de um objeto e o diagrama de atividade que modela a seqüência geral de ações para vários objetos e casos de uso.

Imagine um semáforo de rua. É possível prever três estados para ele: verde, amarelo e vermelho. O diagrama que permite especificar esta transição entre os eventos é o diagrama de estados. A mudança de estado dispara ações diferentes do sistema e que, por sua vez, modificam o estado do objeto (sinal vermelho: ação, parar!).

Já os diagramas de atividade conseguem especificar situações, como paralelismos e sincronizações, que são impossíveis de serem especificadas no diagrama de casos de uso.

SEÇÃO 1 - Modelo de Estados

Os objetos de um sistema modificam seu estado de forma análoga a objetos do mundo real. Pense no cozimento do macarrão. Em seu primeiro estado ele está duro pois não está cozido. Depois de cozido ele irá amolecer e assim entra em outro estado. Esta modificação é chamada de transição entre estados.

Você pode dizer que um estado representa o resultado de atividades executadas por um objeto, determinada pelos valores de seus atributos e pela sua ligação com outros objetos. FURLAN (1998).

Os objetos do sistema passam por vários estados e esta transição faz com o próprio sistema se modifique.

Na verdade, **a transição ocorre porque um evento (mensagens, timer, erros, condições sendo satisfeitas, entre outros) disparado no sistema, faz com que o objeto realize determinadas ações que fazem com que o objeto modifique seu estado.**

O diagrama de estados – DTE deve ser utilizado somente para algumas classes, ou seja, somente para aqueles que possuem estados bem definidos e onde a mudança de estados propicia a mudança do comportamento das classes.

No diagrama a seguir é possível mapear os possíveis estados dos objetos e as ações e condições necessárias para que ocorra a mudança de estados entre os objetos.

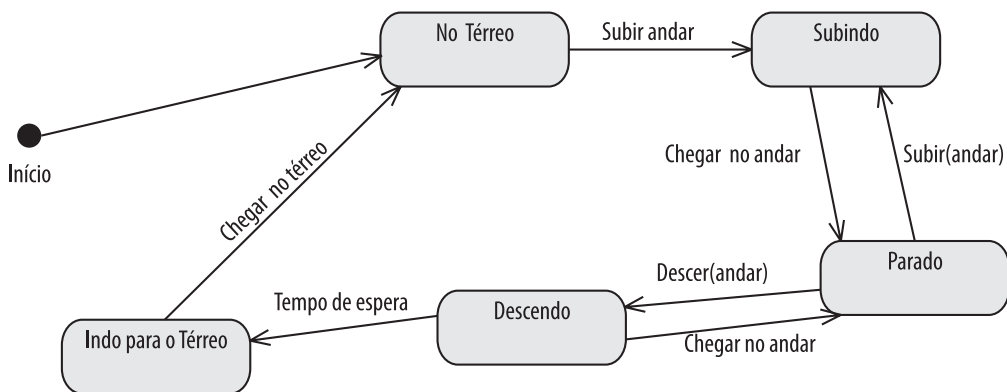



Figura 8.1 - Diagrama de estados Elevador

O DTE desta figura mostra:

- os estados de um objeto;
- os eventos ou mensagens que causam transição;
- as ações que resultam de uma mudança de estado.

Quais são os componentes de um diagrama de estados?

A existência de estado em um objeto indica que a ordem na qual as operações são executadas. No DTE é importante a descrição da ordem das operações no tempo, pois esta ordem pode formalizar a caracterização do comportamento de um objeto.

O DTE utiliza-se de alguns componentes, são eles:

a) Estado - é uma situação na vida de um objeto durante a qual ele satisfaz alguma condição ou realiza alguma atividade em resposta a um evento ou espera a ocorrência de algum evento. No diagrama DTE o estado é representado por um retângulo arredondado.

Emitindo nota / O macarrão está cozido/O menino está nadando

Objetos: nota, macarrão, menino

Estado: emitindo, nadando, cozido

Emitindo Nota

Analisando Produto

Figura 8.2 - Exemplos de Estados

b) Estado Inicial - o estado inicial de um objeto ocorre quando ele é criado, é representado por um pequeno círculo fechado. O estado inicial indica a partir de onde o DTE deve ser lido. Para cada DTE você só tem **um** estado inicial.

Figura 8.3 - Estado Inicial

c) Estado Final - o estado final indica o final do ciclo de vida de um objeto, é representado por um círculo eclipsado. O estado final é opcional e pode existir mais de um, em um mesmo DTE.

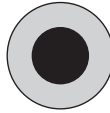



Figura 8.4 - Estado Final

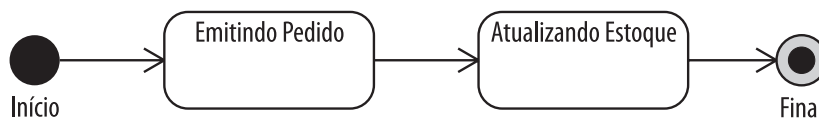



Figura 8.5 - Exemplo DTE

No diagrama da figura 8.5, o estado final acontece logo depois da atualização do estoque que modifica o estado do objeto estoque.

d) Transição – quando a ação ou atividade de um estado está completa, o fluxo de controle passa ao estado seguinte de ação. Especifica-se esse fluxo utilizando transições para mostrar o caminho de um estado de ação ou de atividade para o estado seguinte.

Graficamente você representa a transição por uma linha simples com uma direção. As transições não-ativadas podem ter condições de proteção, significando que a transição será iniciada somente se essa condição for satisfeita.

Imagine quando você está no banheiro pronto para iniciar o banho. Automaticamente você pensa em duas possibilidades: o estado do chuveiro está ligado ou desligado. Em outras palavras: você pode representar isto em um diagrama de estados. Neste DTE o evento é: Girar torneira, e as ações são Abrir e Fechar.

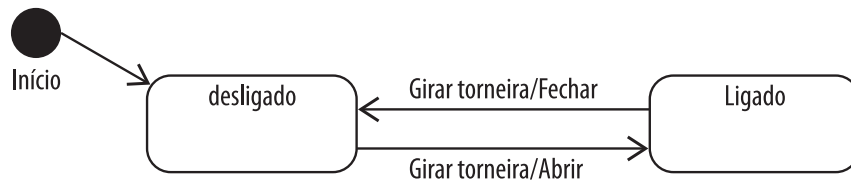



Figura 8.6 - Exemplo DTE tomar banho

Furlan (1998), define quatro possibilidades de eventos em um DTE:

- **Recibo de sinal explícito do outro objeto** – são gatilhos de uma transição (recibo de mensagens). O objeto recebe um sinal de outro objeto e muda de estado. Neste tipo de evento o objeto que envia a mensagem fica esperando a finalização da mesma.
- **Passagem de período designado de tempo** – o evento vai acontecer quando um determinado tempo acontecer disparando a mudança de estados. Neste tipo de evento utiliza-se a cláusula after. Por exemplo: se você tiver o evento after (1 minuto) significa que o evento vai ser executado um minuto depois do objeto entrar no estado atual.
- **Uma condição tornando-se verdadeira** – é mostrada uma condição de guarda em uma transição de estados. Nestes casos você utiliza a cláusula when, por exemplo, when (quantidade < 5). Neste caso, a transição é disparada se a quantidade for menor do que 5.
- **Recibo de chamada de operação pelo próprio objeto ou outro objeto** – é mostrado como uma assinatura de evento em transição de estado. Um objeto solicita um serviço a outro objeto.

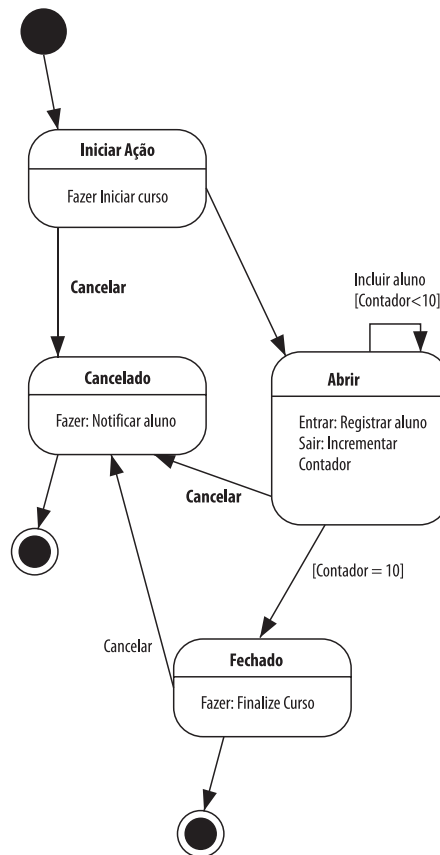



Figura 8.7 - DTE para matrícula de um curso

Fonte: Esmin (2005).

Quando você possui uma ação, a ação possui um processo, a transição do estado. A transição de estado, normalmente é um processo de curta duração e que não pode ser interrompido.

Muitas vezes ao invés de ações, você pode ter atividades relacionadas. Quando isto acontece, você também possui um processo associado, mas o processo está associado a um estado. O processo é mais duradouro e a atividade pode ser eventualmente interrompida por um evento.

A condição de guarda é uma expressão de valor lógico usada em uma transição. Você pode definir a condição utilizando-se parâmetros, referências e ligações da classe.

Observe na figura 8.7: Só será incluido um aluno no curso se contador for menor que 10

Quando você define uma condição de guarda, ela só é disparada se o evento associado ocorrer e se a condição for verdadeira.

A condição de guarda sempre aparece no DTE com o uso de colchetes, como por exemplo:

Realizar_saque (quantia) [quantia=saldo]/sacar (quantia)

No diagrama da figura 8.8, Lima (2004) apresenta um DTE para Pedidos. O pedido pode ser confirmado ou cancelado pelo cliente. O estado Pendente possui uma ação reflexiva Pedidos Pendentes. Ao finalizar o Pedido o estado do Pedido estará alterado.

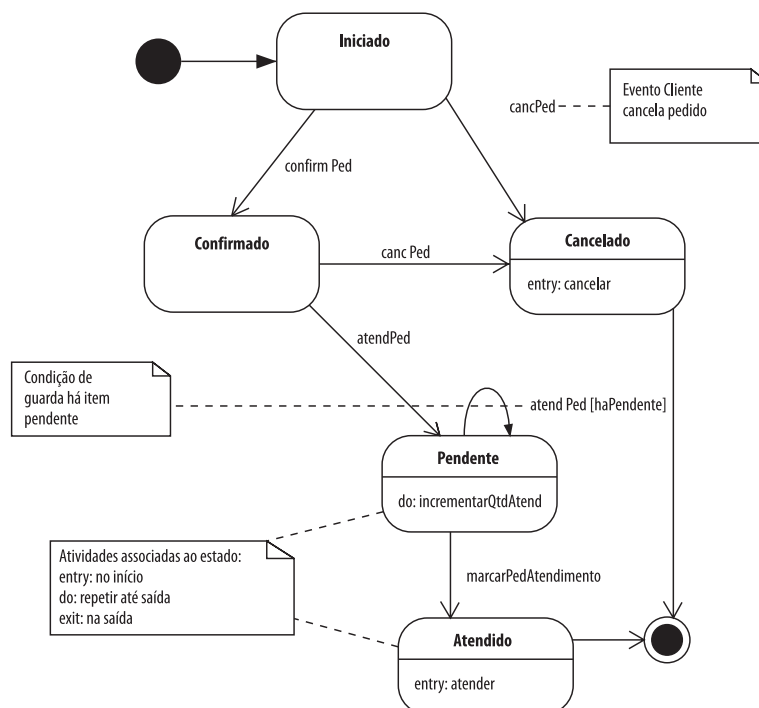



Figura 8.8 DTE Emitir Pedido (LIMA, 2004)

Lembre-se, utilize o diagrama de transição de estados para:

- descrever o comportamento de um objeto ao longo de vários casos de uso;
- modelar objeto dotado de comportamento muito dinâmico.

SEÇÃO 2 - Modelo de Atividade

O diagrama de atividades é o quarto diagrama responsável pela descrição dos aspectos dinâmicos de um caso de uso. O diagrama de atividade modela a lógica utilizada em um caso de uso.

Neste caso, o diagrama de atividade permite apresentar interações, decisões e passos executados em paralelo impossíveis de representar somente com o caso de uso.

Pode ser utilizado para descrever um processo de negócio, a partir da necessidade de entender melhor um problema. O diagrama vai permitir que você entenda melhor o comportamento do sistema, no decorrer dos diversos casos de uso.

Este modelo também é utilizado para especificar a programação com ***multithreading.***

O diagrama de atividade lembra muito o antigo fluxograma, lembra-se dele?

O diagrama de atividade deve ser usado em situações onde todos ou a maioria dos eventos, representam a conclusão das ações geradas internamente, ou seja, os fluxos processuais de controle, bem como em situações onde acontecem eventos assíncronos.

O diagrama permite escolher a ordem pela qual as coisas devem ser feitas, indicando as regras essenciais de seqüência que devem ser seguidas.

O conceito de *multithreading* utiliza o conceito de *thread* que é considerado um processo leve, onde o espaço de endereçamento é compatilhado por vários programas. No ambiente *multithreading* não existe a idéia de programas associadas a processos, mas a *threads*. O processo neste ambiente tem um *thread* de execução, mas compartilha o espaço de endereçamento com inúmeros outros threads. (MACHADO, 2002).

Para FURLAN (1998), os diagramas de atividades podem modelar o sistema de duas maneiras:

- Para modelar o fluxo de trabalho – as atividades são focalizadas conforme visualizadas pelo ator. Por exemplo, no fluxo de trabalho de processamento de um pedido incluirá classes como Pedido e Cobrança. As instâncias dessas duas classes serão produzidas por determinadas atividades (processar pedido criará um objeto Pedido, por exemplo); outras atividades poderão alterar esses objetos (por exemplo, Enviar pedido modificará o estado do objeto Pedido a ser preenchido).
- Para modelar uma operação – os diagramas são empregados como fluxogramas. O diagrama permite visualizar, especificar, construir e documentar o comportamento de qualquer elemento da modelagem. Os diagramas de atividades podem ser anexados a classes, interfaces, componentes, nós, casos de uso e colaboração.

Na figura 8.9, observe os possíveis componentes de um Diagrama de Atividades.

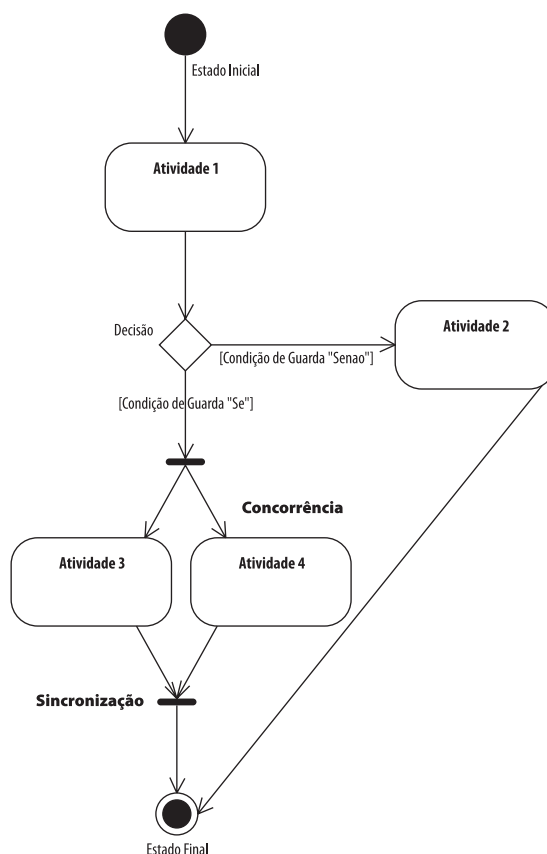



Figura 8.9 - Modelo de um Diagrama de Atividades

O diagrama sempre tem um estado inicial e um estado final. A transição de término liga um estado a outro, ou seja, o término de um passo é o início de outro. Cada uma das ações é disparada no momento em que ocorre o término do evento anterior.

Os pontos de ramificações são pontos onde, a partir de uma transição de entrada, você pode ter várias transições de saída. É o caso das condições de guarda do diagrama. Os objetos apresentados no diagrama podem ser:

- **Atividade** – representada pelo retângulo ovalado.
- **Objeto** – representado por um retângulo.
- **Seta cheia** – relação de precedência entre atividades.
- **Seta pontilhada** – consumo ou produção de objeto por atividade.
- **Linha horizontal cheia** – ponto de sincronização
- **Pequeno círculo cheio** – estado inicial
- **Pequeno círculo eclipsado** – estado final

No diagrama você percebe que existem dois fluxos paralelos (atividade 3 e 4) e que não existe limitação para o número de processos paralelos, pois a sincronização destes fluxos acontece pelo uso de uma barra paralela.

A barra pode ser utilizada para bifurcação ou junção. Se for uma barra de junção (*join*) (no diagrama aparece como sincronização), dois ou mais fluxos de transição serão unidos em um único fluxo. Se for uma barra de bifurcação a partir de uma transição de entrada são criados dois ou mais fluxos paralelos (no diagrama aparece a bifurcação na condição de guarda "se").

Na figura a seguir, você vê o diagrama de atividade para o caso de uso realizar saque do sistema de caixa eletrônico.

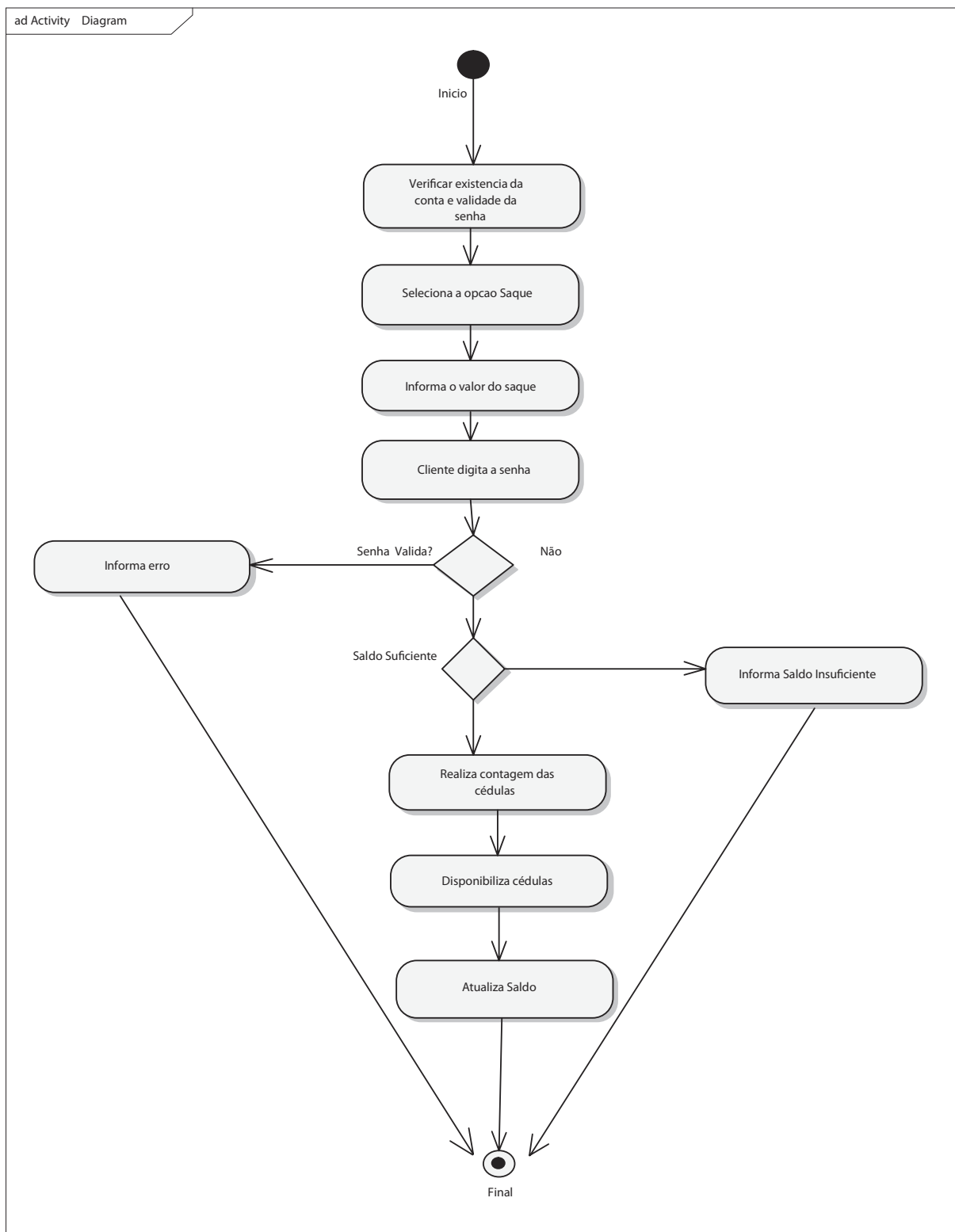



Figura 8.10. Diagrama de Atividades Realizar Saque do Sistema de Caixa Eletrônico

É possível observar pelo diagrama a seqüência de atividades que devem ser realizadas e a precedência de cada atividade.

Na figura a seguir é apresentado o diagrama de atividade para o caso de uso Gerar Contrato de Aluguel do sistema Imobiliário:

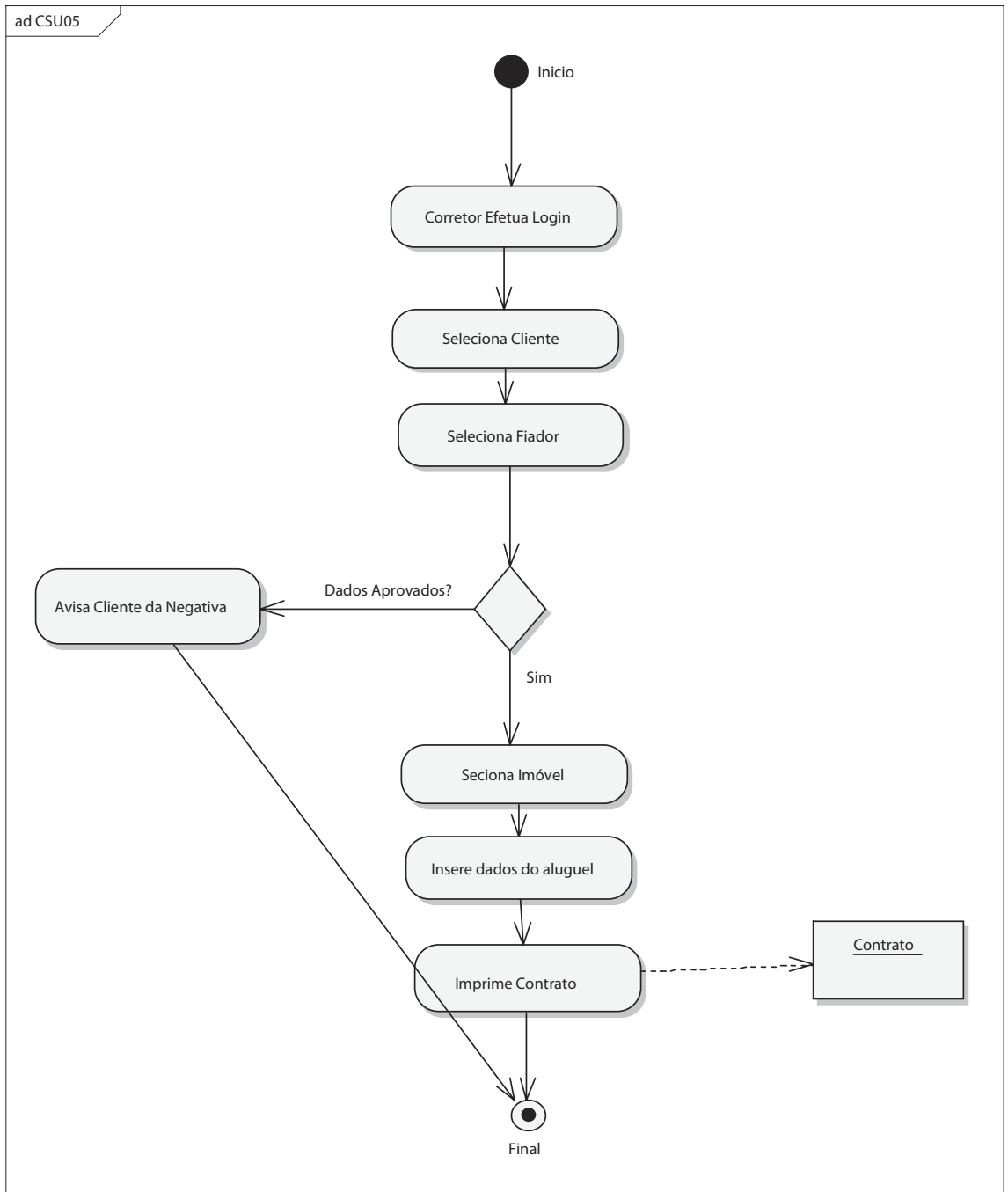



Figura 8.11. Diagrama Gerar Contrato de Aluguel do Sistema Imobiliário

Lembre-se, utilize o diagrama de atividade para:

- Explicitar comportamentos paralelos
- Na modelagem de *Workflow*
- Em caso de programação com *multithreading*
- Para melhor entendimento de *workflow* entre vários casos de uso

Workflow – o *workflow* descreve tarefas dos processos de negócio (conjunto de uma ou mais atividades relacionadas que, coletivamente, atingem um objetivo) em um nível conceitual necessário para compreender, avaliar e reprojetar o processo de negócio. (Bortoli, 2004)

SEÇÃO 3 - Considerações sobre o uso da orientação a objetos

Durante este estudo não foram apresentados todos os diagramas e todas as visões possíveis da UML. Na verdade, foram privilegiadas as visões consideradas fundamentais para qualquer desenvolvimento de *software*.

A complementação pode ser feita por meio da leitura do livro UML Guia do Usuário escrito por Booch, Rumbaugh e Jacobson em 2000.

Ao finalizar esta unidade é importante retomar um tema importante: quais são os benefícios do uso da orientação a objetos? Pode-se afirmar que o uso deste paradigma proporciona o melhor gerenciamento das funções do sistema, o aumento da produtividade pelo uso da reutilização, a melhor qualidade dos serviços oferecidos e a facilidade do mapeamento do mundo real versus o mundo computacional.

Apesar de todas estas vantagens, o mercado ainda não absorveu completamente esta metodologia. Mas, por que?

Furlan (1998) lista alguns motivos para tal quadro:

- Incerteza;
- Falta de mão-de-obra qualificada;
- Ferramentas imaturas;

- O investimento da empresa já realizada em ferramentas não-orientadas a objetos.

A introdução da UML, suas facilidades e o grande número de ferramentas que suportam sua notação vêm gradativamente revertendo este quadro. O grande potencial de comunicação e reaproveitamento pela reutilização de modelos em futuras aplicações, tem aproximado empresas de desenvolvimento do modelo orientado a objetos.

Agora para praticar os conhecimentos conquistados nesta unidade, realize a seguir as atividades propostas.

Atividades de auto-avaliação

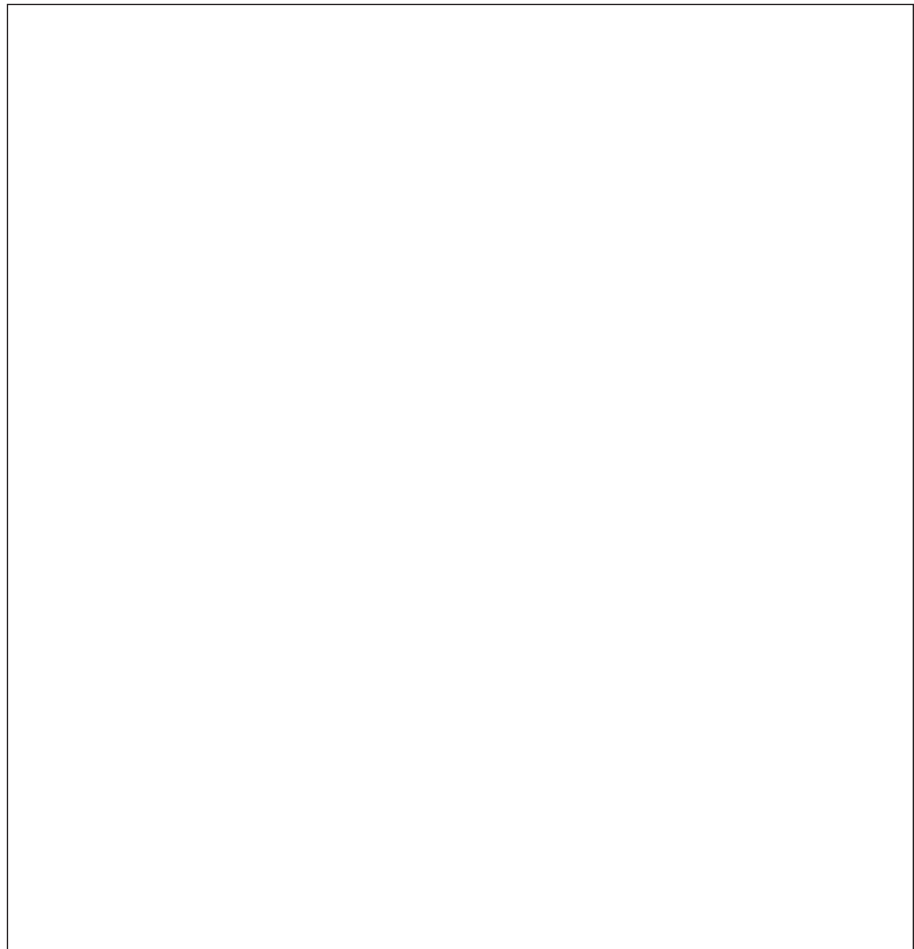
Leia com atenção os enunciados e realize as questões propostas:

1) Complete as afirmações a seguir:

a) Esta modificação é chamada de ________________entre estados.

b) Os_______________ representa o resultado de atividades executadas por um objeto.

c) O _______________indica o final do ciclo de vida de um objeto.

d) Os_______________ de um sistema modificam seu estado de forma análoga a objetos do mundo real.

2) Indique se os conceitos abaixo fazem parte do DTE- diagrama de estados ou do DA- diagrama de atividade:

a) ( ) O diagrama mostra os estados de um objeto.

b) ( ) O diagrama explicita comportamentos paralelos.

c) ( ) O diagrama mostra os eventos ou mensagens que causam uma transição.

d) ( ) O diagrama apóia o entendimento de *workflow* entre vários casos de uso.

3) Relacione os conceitos abaixo, observe que uma mesma opção pode se repetir:

A) After

B) Estado Inicial

C) When

D) Transient

E) Join

a) (   ) É utilizado em um DA para indicar uma bifurcação.

b) (   ) Ocorre quando o objeto é criado.

c) (   ) Utilizado para designar a passagem de um período pré-determinado de tempo.

d) (   ) É utilizado para representar uma condição de guarda.

e) (   ) É utilizado em um DA para indicar uma sincronização.

4) Construa o diagrama de atividades do caso de uso Agendar Horário da clínica Bem-Estar a partir da visão do ator Atendente (veja na unidade 5, atividade 4 de auto-avaliação).

Síntese

Com esta unidade você complementou seu conhecimento sobre a representação dinâmica do sistema por meio de seus diagramas de atividade e estado.

Os fluxos de estado são excelentes na descrição de fluxos complexos, permitindo a visualização da execução do caso de uso.

O diagrama de estado descreve muito bem o comportamento de um único objeto, mas quando o comportamento envolve diversos objetos é mais adequado o uso do diagrama de atividades.

O diagrama de atividades suporta a descrição de comportamentos paralelos modelando o fluxo de trabalho, principalmente quando diferentes casos de uso interagem entre si ou utilizam multiprocessamento. Outra situação de uso comum para o diagrama de atividade é seu uso para entender o comportamento, dependências das ações e as próprias ações necessárias na realização do caso de uso.

Na próxima unidade você vai conhecer de forma mais profunda, a metodologia *Rational Unified Process* – RUP e suas propostas para a melhoria do processo de desenvolvimento de *software*.

Saiba mais

Para aprofundar as questões abordadas nesta unidade você poderá pesquisar em:

- BEZERRA, E. **Princípios de Análise e Projeto de Sistemas com UML**. São Paulo: Campus, 2002. (capítulos 10 e 11).•
- BORTOLI, A. **O uso do *workflow* para Apoiar a Elicitação de Requisitos, *Workshop* em Engenharia de Requisitos**. 2004
- FURLAN, José. **Modelagem de Objetos**. São Paulo: Makron Books, 1998. (capítulo 6).
- BOOCH; RUMBAUGH; JACOBSON. **UML**:Guia do Usuário. São Paulo: 2000. (capítulos 18 e 19).

UNIDADE 9

9

RUP e ICONIX

Objetivos de aprendizagem

- Entender o que é o RUP, seus elementos e conceitos.
- Discutir e analisar nuances do processo de desenvolvimento orientado a objetos utilizando-se o RUP.
- Entender como o RUP colabora para a estruturação efetiva de tarefas e fluxos de trabalho de profissionais, atuando em equipes de desenvolvimento de *software*.
- Compreender o processo ICONIX e seus componentes.

Seções de estudo

Seção 1 Onde se quer chegar?

Seção 2 Quais são as fases do RUP?

Seção 3 Quais os elementos do RUP?

Seção 4 ICONIX

Para início de estudo

Quando você lê um artigo sobre empresas de desenvolvimento de *software* é raro encontrar discussões sobre a dificuldade da empresa em dominar uma tecnologia como um banco de dados, uma linguagem de programação ou mesmo um sistema operacional.

Por outro lado, você vai encontrar dezenas de matérias relatando os problemas relacionados a cronogramas, entregas, produtividade da equipe, qualidade do produto, uso de metodologias e padronização das diferentes etapas.

Ensinar UML a uma equipe de projeto pode não ser uma tarefa tão árdua, como propor a esta equipe todo um processo de desenvolvimento voltado para uma metodologia, utilizando esta notação.

O *Rational Unified Process* é um processo de engenharia de *software* que pretende aumentar a produtividade da equipe, oferecendo práticas eficientes executadas por meio de diretrizes, *templates* e orientações sobre ferramentas para todas as atividades críticas de desenvolvimento de *software*s.

O ICONIX no entanto é um método menos complexo, com um volume menor de documentação que o RUP mas que apresenta grande aceitação no mercado!

Esta unidade vai abordar o método RUP oferecido como uma proposta de solução para os principais problemas relacionados à gerência do processo de desenvolvimento de *software*s.

SEÇÃO 1 - Onde se quer chegar?

Quando olhamos ao redor percebemos nossa total dependência dos sistemas de informação. Mas, o que se faz em nossas vidas modernas que não envolva recursos computacionais?

Poucas são as atividades que nos restam onde não temos como apoio um *software*. Aliada à dependência, temos os riscos envolvidos neste problema complexo. Antigamente se imaginava

arriscado um *software* existente em uma aeronave que ao falhar poderia fazer com que a mesma caísse ou um sistema de controle de uma usina nuclear.

Hoje os riscos extrapolam questões vitais. Imagine os riscos financeiros provenientes das milhares de transações bancárias, os riscos em uma fraude eleitoral ou mesmo os riscos existentes em um *software* de defesa de um país como a Inglaterra.

O desenvolvimento de *software* é uma atividade de alto risco. Entre lucros gerados nesta atividade, muitos foram os prejuízos. Na maioria das vezes pelo mau planejamento, má gerência e baixa qualidade do produto final.

Mas, como gerenciar o processo de desenvolvimento de *software* aumentando sua qualidade se a empresa de desenvolvimento de *software* não conhece seu próprio processo de desenvolvimento?

Segundo Booch (2000), um **processo** é um conjunto de passos parcialmente ordenados com a intenção de atingir metas. A meta neste caso é a entrega eficiente e previsível de um produto de *software* que atenda de forma completa, todas as necessidades do usuário.

O processo define: quem está fazendo o que, quando e como está sendo feito e as ações para atingir uma determinada meta.

Mas como inserir qualidade no processo? A qualidade passa pelo uso de metodologias e reconhecimento formal do processo.

Apenas modelar o sistema não é o suficiente. O uso de uma notação voltada à orientação a objetos como a UML (*Unified Modeling Language*) pode colocar sua empresa na vanguarda em termos de notação, porém pouco garante quanto a qualidade de todo o processo.

A UML é uma linguagem de especificação, o seu uso garante a confecção de diagramas precisos. Mas se estes diagramas não forem usados de forma sistemática, documentados e servirem como pontos de controle e avaliação do processo, de pouco servirão para a adequação de qualidade do produto.

Em resumo, só descrever os diagramas não é o suficiente para garantir um processo de desenvolvimento com qualidade. É

necessário o uso de uma metodologia que unifique o esforço da equipe dentro de um processo formal e mensurável.

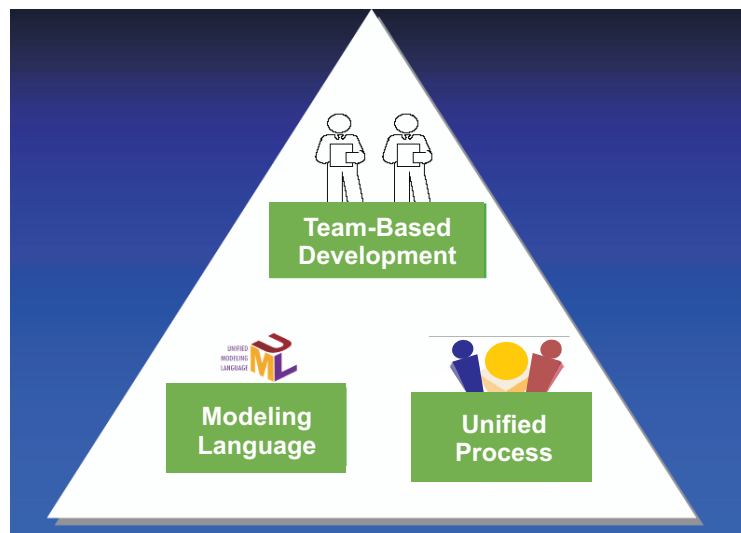



Figura 9.1 - Métodos (IBM, 2005)

Como está estruturado o RUP - *Rational Unified Process*?

O RUP usa a abordagem da orientação a objetos em sua concepção. Sua projeção e documentação utiliza a notação UML para especificar, modelar e documentar artefatos com os quais procura-se ilustrar os processos em ação.

O RUP foi desenvolvido pela *Rational Software Corporation*, sendo então um método proprietário de desenvolvimento de *software*.

Segundo IBM (2005) o RUP se apresenta como um processo de engenharia de *software* que oferece uma abordagem baseada em disciplinas para atribuir tarefas e responsabilidades dentro de uma organização de desenvolvimento, permitindo o gerenciamento eficiente do processo.

Sua meta é garantir a produção de *software* de alta qualidade que atenda às necessidades dos usuários dentro de um cronograma e um orçamento previsíveis (IBM, 2005).

Kroll (2003) apresenta três definições para o *Rational Unified Process*:

- O RUP é uma maneira de desenvolvimento de *software* que é iterativa, centrada à arquitetura e guiada por casos de uso.
- O RUP é um processo de engenharia de *software* bem definido e bem estruturado. O RUP define claramente quem é responsável pelo que, como as coisas devem ser feitas e quando fazê-las. Além disso, também provê uma estrutura bem definida para o ciclo de vida de um projeto RUP, articulando claramente os marcos essenciais e pontos de decisão.

- O RUP é também um produto de processo que oferece uma estrutura de processo customizável para a engenharia de *software*. O produto IBM RUP suporta a customização e autoria de processos e uma vasta variedade de processos, ou configuração de processos. Essas configurações do RUP podem ser criadas para suportar equipes grandes e pequenas, e técnicas de desenvolvimento disciplinadas ou menos formais.

Quais são as diretrizes do RUP?

O RUP apresenta características próprias no processo de desenvolvimento:

- **O uso de um processo iterativo** - o problema e a solução são organizados em pequenas partes, para cada pequena parte do sistema é feita uma iteração. A iteração segue o modelo seqüencial tradicional, com identificação de necessidades, análise, projeto, implementação o desenvolvimento.

- **O processo é incremental** - cada interação acrescenta incrementalmente novas características ao produto. Assim, a cada ciclo a solução delineada é aperfeiçoada.
- **Dirigido por casos de uso** – o processo é guiado pelas interações entre o sistema e usuários. Todos os casos de uso de um sistema compõe a especificação funcional do sistema (modelo de casos de uso), ou seja, definem os requisitos e objetivos do sistema do cliente. Assim os casos de uso associam todos os *workflows* de forma conjunta, dirigindo várias atividades de desenvolvimento como a criação e validação da arquitetura do sistema, a criação de casos de teste, o planejamento das iterações, a criação de documentação do usuário e a implantação do sistema. Os casos de uso sincronizam conteúdo dos modelos criados em cada *workflow*.
- **Centrado na arquitetura** – neste caso, o processo focaliza o desenvolvimento inicial e a linha de base da arquitetura de *software* utilizando-se de relacionamentos claros entre componentes da arquitetura. Você pode ver o sistema, como a soma de diversas partes menores (componentes), cada componente possui a funcionalidade necessária. Sob o ponto de vista da aderência ao negócio ou em termos de implementação, estes componentes se comunicam através de interfaces. O uso de componentes torna a manutenção e a reutilização muito mais eficientes.

Você pode encontrar esse padrão no site do *Rational Software Corporation*: <http://www.rational.com>. Para que você o utilize da forma mais adequada, ele dever ser configurado de acordo com as necessidades da empresa ou mesmo do projeto.

O RUP foi desenvolvido para ser aplicável a qualquer tipo de projeto. Na literatura, é assumido como um *framework* genérico para processos de desenvolvimento.

SEÇÃO 2 - Quais são as fases do RUP?

As fases de um projeto podem ser definidas como o período de tempo necessário entre dois marcos de progresso de um processo, em que um objetivo é alcançado, artefatos são concluídos e decisões sobre a etapa seguinte são tomadas.

Segundo o site: www.rational.com, acessado em 2005, o RUP é composto por quatro fases:

a) Concepção ou Inicial – estabelece o caso de negócio para o projeto. É estabelecido o escopo e a viabilidade econômica do projeto.

b) Elaboração – nesta fase são estabelecidos o plano de projeto e uma arquitetura sólida. Os principais riscos são eliminados. Ao final da etapa é estabelecida a arquitetura, a partir da qual o sistema vai evoluir.

c) Construção – nesta etapa o sistema é desenvolvido. O produto completo é desenvolvido iterativamente.

Para saber

Workflow - É a automação de processos de negócio, em que as atividades são passadas de um participante para o outro, de acordo com um conjunto de regras definidadas.

Framework – No desenvolvimento do *software*, um *Framework* é uma estrutura de suporte definida em que um outro projeto do *software* pode ser organizado e desenvolvido. Tipicamente, um Framework pode incluir programas de apoio, bibliotecas de código, linguagens de *script* e outros *software*s para ajudar a desenvolver e juntar diferentes componentes do seu projeto.

Fonte: Texto disponível no site <http://pt.wikipedia.org>. Acessado em: Ago. 2005.

d) Transição – fornece o sistema aos seus usuários finais, normalmente por uma versão beta do sistema. Se necessário, no final da fase pode ser iniciado um novo ciclo de desenvolvimento para a evolução do produto.

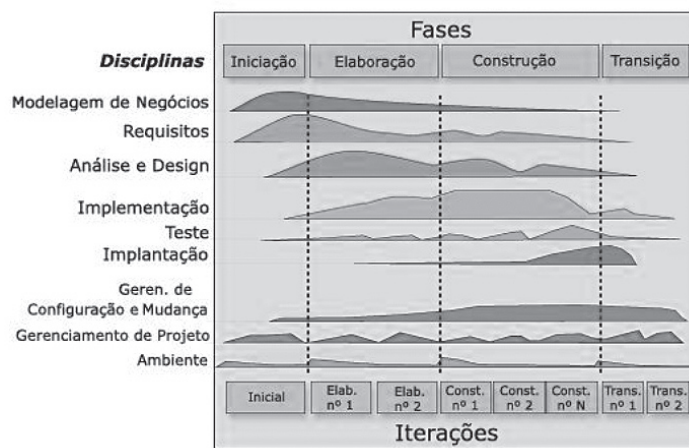



Figura 9.2 - Fases do RUP

Fonte: Adaptação IBM (2005).

A cada fase ocorre um ciclo completo de desenvolvimento, da análise ao produto executável. Cada fase é finalizada com um marco de progresso (*milestone*) onde são verificados se os objetivos da fase foram alcançados.

A cada iteração (elaboração, construção, transição) são realizadas atividades listadas à esquerda do gráfico, as curvas apresentadas no gráfico indicam o esforço dedicado às atividades a cada fase do projeto.

As interações são ciclos completos de desenvolvimento. Cada interação passa por vários fluxos de trabalho do processo com ênfases diferentes.

Como foi comentado na primeira seção, o RUP é direcionado pelos casos de uso. Mas como isto acontece?

A cada iteração são identificados e especificados os casos de uso mais relevantes, é feita então a análise e projeto dos casos de uso, sempre usando a arquitetura como guia. A partir deste ponto são implementados os componentes que realizam o que foi projetado e, finalmente verifica-se se os componentes que satisfazem o que foi previsto como objetivos de cada caso de uso.

Os casos de uso são usados durante todo o processo:

- O caso de uso é usado para especificar os requisitos.
- Na etapa de análise, projeto e implementação os casos de uso são executados.
- Na etapa de testes você vai verificar se o sistema realiza o que está descrito no Modelo de Casos de Uso.
- E, finalmente, os casos de uso são a ferramenta fundamental no planejamento e acompanhamento de todas as iterações.

SEÇÃO 3 - Quais os elementos do RUP?

Segundo BOOCH (2000), quando você utiliza a metodologia RUP, percebe-se imediatamente a existência de cinco elementos principais: os papéis, os artefatos, as atividades, a disciplina e os fluxos de atividade (subdividido em fluxos principais e fluxos de apoio).

a) Papéis

Um papel (ou perfil) define o comportamento e as responsabilidades de um determinado indivíduo ou grupo de indivíduos que trabalham dentro de uma equipe.

É importante que você perceba que os papéis não são indivíduos, mas sim o papel que ele assume no processo. Veja alguns papéis:

- O analista de sistema onde o indivíduo que assume este papel coordena a obtenção dos requisitos, a modelagem dos casos de uso e a definição do escopo do projeto.
- O projetista de testes, responsável por toda a estruturação da etapa de testes, do planejamento à avaliação dos resultados dos testes.

Assim, em um projeto você pode ter vários indivíduos executando um mesmo papel.

b) Atividade

Uma atividade é uma unidade de trabalho que um indivíduo executa quando está exercendo um determinado papel, produzindo um resultado para o contexto do projeto. A atividade é composta de objetivos, passos, entradas e saídas, o responsável pela atividade e os guias e padrões que devem ser seguidos pela atividade.

Exemplo: Uma atividade pode ser a definição dos atores, a definição dos casos de uso ou mesmo conduzir uma etapa de testes.

c) Artefato

O artefato é o produto de um projeto, é o resultado de uma etapa. Apesar de aparecerem como resultado do desenvolvimento do projeto, podem ser utilizados como entrada de uma atividade, e ainda assim no final da atividade vai gerar outro artefato como saída. Normalmente o artefato é baseado em modelos de como devem ser feitos ou mesmo por documentos que possuem já um formato padronizado.

Exemplo: Um artefato pode ser um modelo de caso de uso, um caso de uso, um código fonte, etc.

d) Disciplinas

Uma disciplina mostra todas as atividades que você deve realizar para produzir um determinado conjunto de artefatos. As disciplinas são descritas em nível geral, com um resumo de todos os papéis, atividades e artefatos envolvidos.

Em alguns pontos do projeto você precisa de um detalhamento maior. Neste caso, ocorre o detalhamento da disciplina por meio do fluxo de trabalho.

Um fluxo de trabalho é uma seqüência de atividades que são executadas para a produção de um resultado para o projeto. Fluxos de trabalho podem ser representados por diagramas de seqüência, diagramas de colaboração e diagramas de atividades da linguagem UML.

e) Fluxos de trabalhos

O RUP é desenvolvido baseando-se em nove fluxos de trabalho de processo que podem ser divididos em fluxos principais e de fluxos de apoio definidos em Booch (2000):

Fluxos principais

- **Modelagem do Negócio (*Business Modeling*)** – envolve o entendimento da estrutura e dinâmica da organização cliente, garantindo que clientes, usuários e desenvolvedores tenham a mesma visão da organização para a qual será feito o desenvolvimento.
- **Requisitos (*Requirements*)** – Esta disciplina visa estabelecer e manter concordância com os clientes e outros envolvidos, sobre o que o sistema deve fazer utilizando-se para isto dos casos de uso. Também é definida a delimitação do sistema, são definidas as bases para planejamento do conteúdo técnico das iterações, estimativas de custo e o tempo de desenvolvimento do sistema e a definição da interface do usuário com o sistema, focando nas necessidades e metas dos usuários.
- **Análise e Projeto (*Analysis and Design*)** – envolve a tradução dos requisitos numa especificação que descreve como implementar o sistema, adaptando o design para que corresponda ao ambiente de implementação, projetando-o para fins de desempenho. É neste momento que são descritas as diferentes visões do sistema.
- **Implementação (*Implementation*)** – envolve o desenvolvimento de código: classes, objetos, etc., teste de unidades e integração de subsistemas.

- **Teste (***Test***)** – descreve todos os casos de teste, procedimentos e medidas para o acompanhamento dos erros ocorridos.
- **Entrega (***Deployment***)** – abrange a configuração do sistema a ser entregue (empacotamento, distribuição, instalação, treinamento de usuários, planejamento e condução de beta testes). São descritos três modos de implantação de produto: a instalação personalizada, o produto em uma forma "compacta" e o acesso ao *software* por meio da internet.

Fluxos de Atividades de Apoio

- Gerência de Projeto (*Project Management*) – Enfatiza principalmente, o gerenciamento de risco, o planejamento de um projeto iterativo por meio do ciclo de vida e de uma iteração particular. O monitoramento do progresso de um projeto iterativo e o uso de métricas. Aspectos relacionados à gerência de pessoal, contratos e orçamento não são observados.
- Gerência de Configuração e Mudanças (*Configuration and Change Management*) – O fluxo controla as modificações mantendo a integridade dos artefatos do projeto. Envole a identificação dos itens de configuração, a restrição de mudanças, a auditoria das mudanças feitas e a definição e o gerenciamento das configurações desses itens.
- Ambiente (*Environment*) – envolve a infra-estrutura necessária para que o desenvolvimento ocorra. A meta é oferecer à organização o ambiente de desenvolvimento de *software* processos e ferramentas que dará suporte à equipe de desenvolvimento.

Quais são os modelos do RUP?

Quais os Prós e Contras do *Rational Unified Process?*

A discussão sobre vantagens e desvantagens do RUP é bastante acirrada.

Uma grande vantagem a ser considerada é o uso de princípios de engenharia de *software* na sua abordagem de desenvolvimento iterativa, incremental, orientada a requisitos e baseada em arquitetura. O sistema desenvolvido com o RUP tende a tolerar melhor mudanças de requisitos e alterações no produto.

O uso de componentes desenvolvidos ao longo de um projeto podem ser reutilizados em outros projetos, diminuindo o tempo de desenvolvimento e, conseqüentemente, os custos para o cliente.

Um dos aspectos mais importantes é a capacidade de gerência por meio de fases e *millestones* incorporados ao projeto.

Ao lado de suas vantagens também temos limitações do método, entre elas, o fato do RUP não abordar a gestão de pessoal e contratos, como o fato da ferramenta ser um sistema de hipertexto dificultando as integraçoes com outras ferramentas.

Talvez um dos fatores mais críticos do método seja o fato do método ser previsto para qualquer porte de empresa. A implantação em empresas de pequeno porte tem se mostrado,no entanto, difícil pelo volume de responsabilidades e a quantidade de atividades de cada fluxo de atividade.

Quer conhecer mais?

Para aprofundar seus conhecimentos sobre esta unidade, acesse a midiateca no EVA. Leia o texto: **Um Método de Desenvolvimento de Sistemas de Grande Porte Baseado no Processo RUP**. Com ele, você vai poder avaliar melhor o processo RUP.

SEÇÃO 4 - ICONIX

O ICONIX é uma metodologia de desenvolvimento de software com características interativas e incrementais. Classificar o ICONIX é difícil, por um lado possui uma veia tradicional com um processo bem definido, por outro lado aproxima-se dos métodos ágeis procurando a redução da documentação e a simplicidade no processo.

Mastelari (2004) define ICONIX Software Engineering como um processo enxuto e robusto voltado para o trabalho com equipes pequenas e desenvolvimentos de tamanho pequeno e médio.

O processo ICONIX teve seu início muitos anos antes da concepção da UML e do processo unificado. Foi elaborado por Doug Rosenberg e Kendall Scott Poe voltado-se a uma abordagem orientação a objetos. Silva & Videira (2001), apresentam o ICONIX como uma metodologia prática, intermediária entre a complexidade do RUP (Rational Unified Process) e a simplicidade do XP (Extreme Programming). A UML – Unified Modeling Language é usada integralmente no método suportando e respondendo questões impostas pela metodologia por meio de seus diagramas.

São, segundo Borillo (2000), três as características fundamentais no ICONIX:

- o modelo é **Iterativo e incremental** e portanto várias iterações acontecem da definição do modelo de domínio à identificação dos casos de uso.
- **Rastreabilidade**: todos os passos do processo referenciam os requisitos, o modelo permite então verificar em todas as fases se os requisitos foram atendidos. Desta forma, pode-se determinar qual o impacto que a alteração de um requisito tem em todos os artefatos do sistema.

- **Aerodinâmica da UML**: a metodologia incorpora o uso da UML por meio de diagramas. Os mais usados são: os diagramas de casos de uso, diagramas de seqüência e colaboração, diagramas de robustez e diagramas de classes.

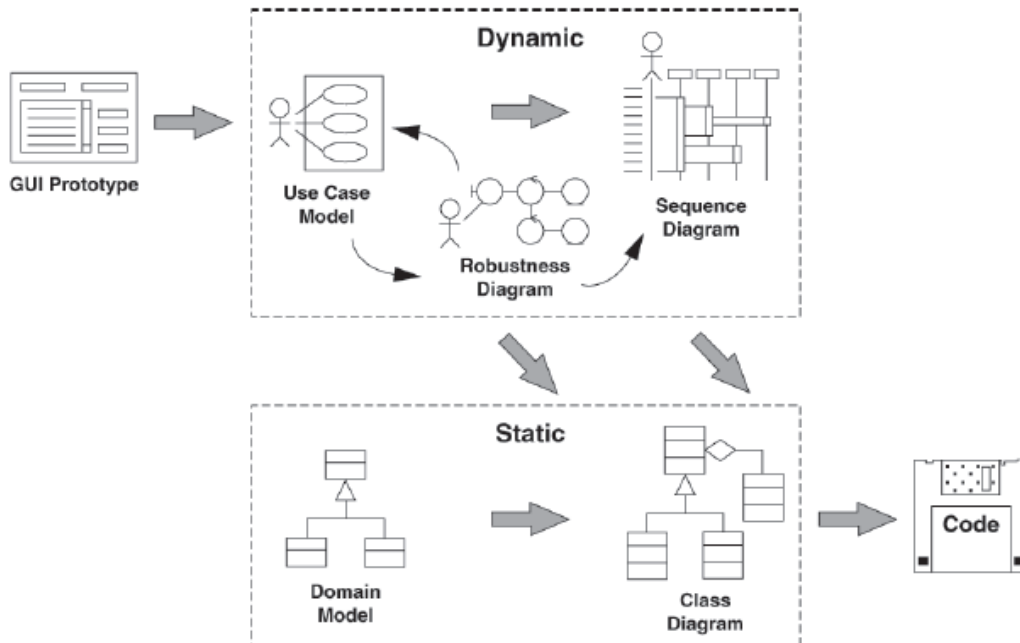



Figura 9.3. Processo Iconix (Rosenberg et all, 2005)

O ICONIX fundamenta-se em dois modelos o estático e o dinâmico. O modelo estático modela o funcionamento do sistema sem se preocupar com interações e aspectos dinâmicos do sistema. São dois os diagramas usados nesta representação: o diagrama de domínio e o diagrama de classe. O modelo dinâmico apresenta as interações do sistema mostrando a interação do usuário com o sistema. O modelo dinâmico é representado por diagramas de seqüência, de casos de uso e o diagrama de robustez.

O ICONIX determina um ciclo com quatro etapas bem determinadas, a análise de requisitos, a análise e projeto preliminar, o projeto e a implementação.

Análise de requisitos

Na análise de requisitos ocorre a identificação das necessidades do cliente por meio dos requisitos funcionais. Nesta fase o contato com o cliente é estreito, segundo Silva et all (2001) a tarefa de análise de requisitos consiste em realizar as seguintes tarefas:

- Identificar os objetos do domínio do problema, nesta identificação utiliza-se o diagrama de classes;
- Desenvolver protótipos da interface, diálogo e navegação proporcionando para o cliente o melhor entendimento sobre a proposta;
- Identificar os casos de uso e os atores associados a estes casos de uso. Esta tarefa será realizada por meio dos diagramas de casos de uso;
- Finalizada a identificação dos casos de uso, os mesmos devem ser organizados por meio de grupos que permitam sua melhor compreensão e visualização. Esta estruturação ocorre por meio do diagrama de pacotes.
- Os requisitos funcionais devem ser claramente associados aos casos de uso e aos objetos do domínio de forma a tornar visível qual caso e classe irá solucionar o requisito funcional.

Análise e Projeto Preliminar

Durante a etapa de análise e projeto são descritos os casos de uso identificando seus cenários. Nesta etapa do processo são construídos os diagramas de robustez e são refinados e finalizados os diagramas de classe.

Projeto

A etapa de projeto permite a equipe a especificação do comportamento esperado nos casos de uso, a identificação dos objetos, atores e as mensagens trocadas entre os elementos. Para esta tarefa o diagrama de seqüência torna-se essencial . Neste momento já é possível inserir no diagrama de classes seus atributos e métodos. São assim concluídos os diagramas do modelo estático validando se todos os requisitos prescritos foram atendidos.

Implementação

Para a etapa de implementação a equipe apresenta seu maior esforço na geração do código, na realização de testes de unidade, integração e aceitação do cliente.

Agora, para praticar os conhecimentos conquistados nesta unidade, realize a seguir as atividades propostas.

Atividades de auto-avaliação

Leia com atenção os enunciados e realize as questões propostas:

1) Assinale as afirmativas corretas (mais de uma caso necessário):

a) ( ) O RUP utiliza-se do modelo iterativo para o desenvolvimento do *software*, isto significa a definição claras de etapas em um ciclo rígido e formal.

b) ( ) O RUP é visto como um produto de processo de engenharia customizável.

c) ( ) O RUP é centrado na construção do produto, baseia-se fundamentalmente no uso de uma linguagem orientada a objetos.

d) ( ) O RUP é centrado na arquitetura, estabelecendo de forma clara e segura todos os relacionamentos existentes entre seus componentes.

2) Relacione os elementos do RUP:

A. Papéis

B. Atividades

C. Artefatos

a) ( ) O analista responsável pelo gerenciamento da qualidade do produto.

b) ( ) Relatório de testes.

c) ( ) Responsável pelo controle da documentação e padronização da equipe de implementação (programas).

d) ( ) Cronograma.

e) ( ) Definir as classes do projeto.

f) ( ) Entender a estrutura e a dinâmica da organização.

3) Relacione os fluxos de atividades do RUP:

A. Modelo de negócios
B. Requisitos
C. Análise e Projeto
D. Gerência de Projeto
E. Gerência de Configuração e Mudanças

a) ( ) Procura manter a integridade dos artefatos do projeto.
b) ( ) Procura manter uma visão uniforme sobre o projeto para todos os envolvidos.
c) ( ) Descreve as diferentes visões do sistema.
d) ( ) Enfatiza o gerenciamento de riscos.
e) ( ) Neste fluxo é definida a delimitação do sistema.

4) Dos nove modelos oferecidos pelo RUP, defina 3 que você considera fundamentais para que o projeto seja bem aceito pelo cliente final.

Síntese

Na décima unidade você teve contato com o RUP e o ICONIX, seus conceitos e principais modelos.

Foi possível perceber a importância do processo incremental iterativo oferecido pelo modelo e a preocupação de tornar a gerência e a manutenção do produto mais eficiente em todas as etapas do desenvolvimento. Você percebeu que dentro do método é fundamental a definição clara dos papéis e responsabilidades. Um dos pontos fortes do modelo é a padronização e a definição de artefatos para cada etapa do projeto.

A utilização de uma arquitetura baseada nos casos de uso, aproxima o usuário final do desenvolvimento, melhorando a qualidade do processo refinando em etapas bastante iniciais o processo de validação do sistema.

Apesar das controvérsias relacionadas a seu uso, o RUP tem-se mostrado uma metodologia eficiente no mercado sendo que, sua utilização vem crescendo gradativamente. Um dos fatores mais fortes para isto é a possibilidade de adaptação do modelo às necessidades e exigências da empresa.

Um ponto forte do ICONIX é a identificação e representação do modelo de domínio e dos casos de uso. A partir deste ponto o processo de desenvolvimento passa a ser iterativo e incremental. Os casos de uso são documentados e a ele é anexado o respectivo diagrama de robustez. Na seqüência identificam-se novos objetos e detalhes que são incorporados ao diagrama de domínio. Na etapa seguinte os diagramas de seqüência são feitos procurando a identificação de objetos. Na última etapa operações são adicionadas assim como os atributos ao diagrama de classe.

O ICONIX é uma sugestão de processo de desenvolvimento de software e tem tido uma grande aceitação no mercado, parte deste sucesso se deve a forma sucinta com que trata a documentação. Outro aspecto relevante é a importância do modelo na etapa de entendimento dos requisitos do cliente. Esta preocupação tem transformado o modelo em um modelo de sucesso.

Saiba mais

Para aprofundar as questões abordadas nesta unidade, acesse a midiateca.

Uma outra metodologia bastante aplicada em empresas de *software* atualmente é o Iconix ((link ICONIX_guj.pdf). No artigo escrito por José Anízio Maia você vai perceber que é uma metodologia simples e menos burocrática do que o RUP. Vale a pena conferir!

Para concluir o estudo

Com este primeiro livro você aprendeu os conceitos e particularidades da análise estruturada e da análise essencial.

Conheceu das análises, características e particularidades de implementação que tornam o projeto mais claro, facilitando sua compreensão inclusive para o usuário final.

O uso das metodologias torna o trabalho do gerente mais objetivo e sua cobrança perante a equipe mais eficiente. Isto se deve pela possibilidade de geração de diferentes tipos de artefatos que podem ser utilizados como marcos de projeto para acompanhamento da equipe.

A análise estruturada foi pioneira em termos de aceitação como metodologia de documentação de projetos. Seu uso ainda hoje é expressivo pela facilidade de utilização de sua notação e benefícios advindos de seu uso.

O segundo livro da disciplina dará ênfase à metodologia orientada a objetos fazendo uso da linguagem de notação Linguagem de Modelagem Unificada. Além de explorar seus diagramas mais utilizados, também será feita uma breve abordagem sobre o *Rational Unified Process* – RUP.

Durante todo o seu estudo nessa disciplina você foi apresentado a três paradigmas diferentes: a análise estruturada, a análise essencial e a análise orientada a objetos.

Você conheceu características e particularidades de implementação de cada uma. Foi possível perceber que a orientação a objetos encaixa-se muito bem no paradigma atual de implantação, mas que todas elas, sem exceção, colaboram para que o projeto torne-se claro para toda a equipe de projeto.

O uso das metodologias torna o trabalho do gerente mais objetivo e sua cobrança perante a equipe se torna mais objetiva. Isto se deve pela possibilidade de geração de diferentes tipos de artefatos que podem ser utilizados como marcos de projeto para acompanhamento da equipe.

Finalizando o estudo, esperamos que o estudo da disciplina tenha lhe proporcionado a oportunidade de reconhecer qual a metodologia mais adequada para seus projetos assim como o entendimento sobre seus conceitos e diagramas.

Prof. Vera Schuhmacher

Referências

AMBLER, S. W. **Análise e Projeto Orientados a Objetos**. Rio de Janeiro: Editora Infobook, 1998.

ARAGÃO, S. **Modelagem Visual de Objetos UML**. São Paulo: [s.e.], 2005.

BECK, K. **Programação Extrema Explicada**. Bookman, 1999

BONA, Cristina. **Avaliação de Processos de Software:** Um Estudo de Caso Em XP e Iconix. Dissertação de Mestrado, Programa de Pós-Graduação em Engenharia de Produção da Universidade Federal de Santa Catarina. 2002.

BOOCH, G. ; RUMBAUGH, J. ; JACOBSON, I. UML **Guia do Usuário**. São Paulo: Ed. Campus, 2000.

BORTOLI, A. **O uso do workflow para apoiar a elicitação de requisitos**. Workshop em Engenharia de Requisitos. 2004.

CHEN, Peter. **Gerenciando Banco de Dados**: a abordagem entidade-relacionamento para projeto lógico. São Paulo: McGraw-Hill, 1990.

COAD, Peter; YOURDON, Edward. **Análise baseada em objetos**. Rio de Janeiro: Campus, 1992.

COMUNIDADE IBM RUP (2005). Disponível em: **http://www-130.ibm.com/developerworks /rational/community**. Acesso: Ago, 2005.

DEMARCO, Tom. **Análise estruturada e especificação de sistema**. Rio de Janeiro: Campus, 1989.

ESMIN ,A. A. A . **Modelando com UML** : Unified Modeling Language,

FILGUEIRA, J.M.; COSTA, W.S. **A Importância de Utilizar Uml para Modelar Sistemas**: Estudo de Caso. Disponível em: <http://www.cefetsp.br/edu/sinergia/6p10c.html>. Acesso em: Ago, 2005.

FOWLER, M. UML ***Distilled Second Edition*****:** *a brief guide to the standard object modeling language*. New Jersey: Prentice-Hall International, 1997.

FURLAN, J.D. **Modelagem de Objetos Através da UML**. São Paulo: Makron Books, 1998.

GILLEANES, T. A. Guedes. **UML**: uma abordagem prática. Novatec, 2004.

IBM RATIONAL UNIFIED PROCESS Versão 2003.06.12.01 . (2005) **http://www-306.ibm.com/***software***/awdtools/rup/**

IBM RUP (2004) **http://www-130.ibm.com/developerworks/rational/products/rup**, novembro.

KROLL, P.; KRUCHTEN P. ***The Rational Unified Process Made Easy***: a *Practitioner's Guide to the RUP"*, Addison Wesley, 2003.

LARMAN, Craig. ***Applying UML and Patterns***: *An Introduction to Object-Oriented Analysis and Design and the Unified Process*. Prentice Hall, 2001.

LIESENBERG, H. **Diagramas de Colaboração**. Disponível em: <http://www.unicamp.br/~hans/mc426/#program>. Acesso em: Ago, 2005.

NIEDERAUER Mastelari. **Contribuição ao Processo de Integração de Informações da Manufatura para Empresas de Pequeno e Médio Porte**. Dissertação (Pós-Graduação em Engenharia Mecânica) Universidade Estadual de Campinas / Faculdade Engenharia Mecânica, 2004.

PACHECO, R. MONTENEGRO, F. **Orientação a Objetos em C++**. São Paulo: Ed. Ciência Moderna, 1994.

PADUA, W.P. **Engenharia de** *Software*. LTC Livros Técnicos e Científicos, 2001.

PAGES-JONES, M. **Fundamentos do Desenho Orientado a Objetos**. Makron Books: 2001.

PAULA FILHO, Wilson de Pádua. **Engenharia de** *software*. São Paulo: Campus, 2001. BEZERRA, E. Princípios de Análise e Projeto de Sistemas com UML. São Paulo: Campus, 2002.

PAULA FILHO, Wilson de Pádua. **Engenharia de** *software*: fundamentos, métodos e padrões. Rio de Janeiro: LTC, 2001.

PAULA, W.P. **Engenharia de** *Software*. São Paulo: LTC, 2001.

PETERS, J. F.; PEDRYCZ, W. **Engenharia de** *software*: teoria e prática. Rio de Janeiro: Campus, 2001.

PRESSMAN, Roger. **Engenharia de** *software*. São Paulo: McGraw-Hill, 2002.

SCHWABER, K. e BEEDLE, M. ***Agile Software Development with SCRUM***. Prentice-Hall, 2002.

SILVA, A.M.R., VIDEIRAM C.A.E., UML. **Metodologias e Ferramentas Case**. Lisboa: Centro Atlântico, 2001.

SOARES, M. S. **Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software**. RESI - Revista Eletrônica de Sistemas de Informação. Edição 4: Ano III, Volume III, Número I. Novembro de 2004.

SOUZA, Francisco Flavio de; BRAGA, Rosana Teresinha Vaccare. **Um Método de Desenvolvimento de Sistemas de Grande Porte Baseado no Processo RUP.** In: 1º SIMPÓSIO BRASILEIRO DE SISTEMAS DE INFORMAÇÃO. Porto Alegre, RS. Anais do 1º SBSI. 2004. p. 31-38.

SUTHERLAND, Jeff, ***Agile Development***: *Lessons Learned from the First Scrum. Cutter Agile Project Management Advisory Service: Executive Update*. 2004.

SOMMERVILLE, Ian. **Engenharia de** *software*. 6. ed. São Paulo: Addison-Wesley, 2003.

TONSIG, Sérgio Luiz. **Engenharia de** *software*: análise e projeto de sistemas. São Paulo: Futura, 2003.

YOURDON, Edward. **Análise Estruturada Moderna**. Rio de Janeiro: Campus, 1992.

WUESTEFELD, Klaus. ***Xispê – Extreme Programming***. 2001. Disponível em: <http://www.xispe.com.br/index.html>. Acesso em: 18 mar. 2002.

Sobre a professora conteudista

Vera Rejane Niedersberg Schuhmacher é Mestre em Engenharia de Produção com ênfase em Usabilidade de *Software* pela Universidade Federal de Santa Catarina-UFSC. Professora da Unisul desde 1998, em disciplinas oferecidas nos cursos de Ciência da Computação e Sistemas de Informação e Pós-graduação . Pesquisadora do Núcleo de Computação, atua como Coordenadora do NPU- Núcleo de Pesquisas em Usabilidade prestando consultoria em Engenharia de *Software* e Usabilidade em empresas de tecnologias e projetos financiados por órgãos de fomento como Finep, CNPq e Funcitec.

Respostas e comentários das atividades de auto-avaliação

A seguir são apresentadas respostas curtas e comentários sobre as atividades de auto-avaliação apresentadas durante as unidades. Para melhor aproveitamento de seus estudos realize a sua conferência somente depois de realizar as atividades propostas

Unidade 1

1) a) V

b) V

c) V

d) F

e) V

2) Seqüência correta: G, D, C, B, E, F, A, D.

3) A afirmativa correta é (b).

4) a) E

b) P

c) I

d) C

5) Alternativa correta é: a.

6) Alternativa correta: c.

O numero de funcionalidades é bastante pequeno, por outro lado as regras de negócio sugerem uma certa confusão para o bom entendimento do analista, o que torna apropriado o uso do modelo prototipação para a identificação de requisitos.

7) Observe o modelo XP e o modelo SCRUM, e a seguir descreva o que é possível determinar como diferenças fundamentais em relação aos modelos tradicionais.

São citadas abaixo características existentes em ambos os modelos:

Tanto o Scrum como o XP apontam como de grande importância a comunicação entre a equipe e seus colaboradores, isto se evidencia por meio de reuniões formais e informais.

Outro aspecto bastante diferenciado está relacionado ao volume de documentação dos projetos, em ambos os modelos procura-se racionalizar evitando o excesso.

Um terceiro aspecto é a participação do usuário de forma efetiva no processo de desenvolvimento.

Os modelos são apontados como ideais para equipes pequenas e com interações rápidas.

Unidade 2

1) Alternativa correta: b.

2) Seqüência correta é: d, a, c, b.

3) Relatório de Análise do Problema

Observe que foram acrescentadas informações com intúito de mostrar os ítens de forma mais completa.

1 - Nome da Empresa: Clínica Bem-Estar

2 - Contato: Sr. Julibio Ritz (gerente) – fone : 3339090 Cel- 9987878

3 - Descrição do problema

A clínica possui 34 médicos cadastrados em diferentes especialidades e presta atendimento a pacientes conveniados aos planos Bruxtr, Vpfzm e UIOlk ou particular.

A clínica funciona com um pequeno número de atendentes responsáveis pela marcação de consultas, preenchimento inicial de dados cadastrais Cada médico faz 3 plantões semanais de 4 horas seguidas, as consultas possuem um intervalo de 30 minutos. Existe a possibilidade da consulta ser de retorno, neste caso são apenas 15 minutos.

A clínica é 24 horas. Cada médico possui uma agenda preta onde sao marcadas as consultas. Na marcação da consulta é colocado o nome do paciente, horário e convênio.

A clínica possui 2 atendentes que são responsáveis por preencher o cadastro inicial do paciente que contem nome, endereço, telefone, data de nascimento, convênio.

O médico ao atender o paciente preenche sua ficha manualmente, informando peso, altura, idade, motivo da consulta, queixa principal, doenças anteriores, diagnóstico, prescrição. A prescrição pode ser a solicitação de exames ou medicamentos com posologia.

4 - Identificação do principal objetivo do cliente.

O objetivo principal é o aumento na eficiência e tempo de resposta no processo de marcação de consultas.

5 - Descrição dos usuários do sistema

Médicos – todos os 34 médicos possuem especialização, todos possuem conhecimentos básicos de informática e noção do funcionamento e procedimentos da clínica. A faixa etária se encontra entre 30 e 48 anos. Nenhum médico possui qualquer deficiência física.

Atendente – as 2 atendentes possuem o segundo grau completo, razoável experiência em informática e conhecimento do processo de funcionamento da clínica. As atendentes estão na clínica a aproximadamente 3 anos.

Paciente – existem aproximadamente 800 fichas na clínica, a clínica não sabe precisar se sua clientela possui acesso a internet. A clínica possui aproximadamente 85% das consultas realizadas por convênio.

6 - Descrição detalhada dos processos existentes (COMO O SISTEMA ATUAL FUNCIONA?)

Marcação da Consulta - O paciente pode realizar o agendamento da consulta pessoalmente ou por telefone, em qualquer dois dos métodos os procedimentos são idênticos.

O paciente solicita a consulta informando o nome do médico ou a especialidade desejada, posteriormente informa a data desejada. A Atendente verifica a possibilidade de marcação da consulta (observando se o médico em questão possui horários vago para a data desejada). Se existe horário disponível a atendente solicita ao paciente o tipo de convênio ou se é particular. Se for convênio é verificado se é um convênio válido, se for particular é informado o valor da consulta. A atendente atualiza a agenda com o nome do paciente e o tipo de consulta(convênio/particular). O tempo para cada consulta é de 20 minutos ou 15 minutos para retorno. O médico possui intervalo de 10 minutos.

A consulta pode ser uma consulta de retorno, neste caso a atendente verifica se a data está ainda dentro do prazo de retorno de 15 dias. Em caso afirmativo, a consulta é marcada na agenda.

Médico Indisponível – Caso o médico solicitado esteja indisponível a atendente procura informar o nome de outro médico disponível naquele horário ou o próximo horário disponível.

Atendimento – se o paciente já possui cadastro o mesmo é convidado a adentrar no consultório do médico. A partir deste momento o médico solicita informações procedimentais para o futuro diagnóstivo preenchendo a ficha do paciente.

Finalizada a consulta o paciente é liberado e a ficha é recolhida pela atendente sendo novamente arquivada.

Se o paciente for novo a atendente solicita o preenchimento da ficha cadastral com dados pessoais.

7 - Itens produzidos no sistema (quais são os relatórios e consultas existentes ou solicitados pelos clientes)

Neste momento a clínica possui os seguintes documentos:

Agenda – são anotados o nome do paciente (60 caracteres), horário (hora/minuto) e convênio (Bruxtr, Vpfzm e UIOlk ou particular).

Ficha do Paciente - nome(60 caracteres), endereço(60 caracteres), telefone(12 caracteres), data de nascimento(date), convênio (Bruxtr, Vpfzm e UIOlk ou particular).

Ficha Médica – apresenta peso do paciente (numérico (03V2), altura (numérico (2V2), idade (numérico 3), motivo da consulta(Alfanumérico 100), queixa principal(Alfanumérico 100), doenças anteriores(Alfanumérico 150), diagnóstico(Alfanumérico 150), prescrição (Alfanumérico 200). A prescrição pode ser a solicitação de exames ou medicamentos com posologia.

Relação de horários médicos – a listagem contem nome do médico ((Alfanumérico 50), Data (date) e horário (numérico 2V2) de atendimento do médico na clínica.

Relação de atendimentos por tipo – o relatório é emitido mensalmente apresentando o nome do convênio listando a partir dele o nome do médico e o total de consultas realizadas para o convênio. Ao final é apresentado o total de atendimentos por convênio.

Obs -3V2 significa uma variação numérica de ate 999,99.

8 - Volume de informações do sistema atual

A clínica possui aproximadamente 800 pacientes cadastrados, 34 médicos ativos e 2 atendentes. Apresenta convênio médico com 3 empresas mas apresenta possibilidades de aumentar este número. A clínica realiza em torno de 56 consultas dia.

9. Descrição de situações consideradas críticas e atores envolvidos

A clínica apresenta situações criticas relacionadas a marcação incorreta de horário, com médico indesejado ou mesmo data e horário indesejado. As atendentes poderiam ser orientadas para telefonar ao paciente confirmando a consulta com três horas de antecedência.

10 - Restrições do projeto

O cliente não deseja dispender recursos com a plataforma de sistema operacional e o banco de dados, sendo que deve ser considerada uma possibilidade *open source*.|

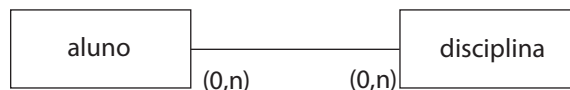
Unidade 3

1) Seqüência correta é: B, G, D, F, C, E, D

2) a) Um professor leciona várias disciplinas em sua universidade

b) A universidade emprega vários funcionários

c) Os funcionários são lotados em um departamento

d) Um aluno pode estar matriculado em nenhuma ou várias disciplinas.

UNIDADE 4

1) Alternativas corretas: a e b.

2) A seqüência é:

a) C

b) O

c) O

d) O

e) C

3) A seqüencia correta é:

a) Poliformismo

b) Encapsulamento

c) Mensagem

d) Herança

4) A seqüencia correta é:

a) A

b) B

c) D

d) C

e) A

f) E

g) B

UNIDADE 5

1) A afirmativa correta é: b.

2) A seqüência correta é:

a) V

b) F

c) F

d) V

e) F

f) V

3) A seqüência correta é:

a) A

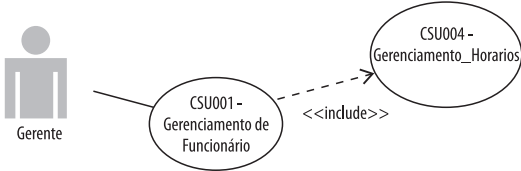
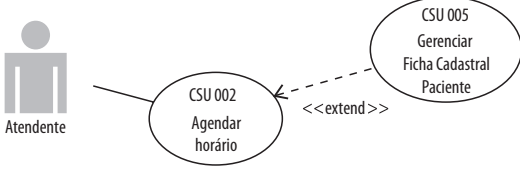
b) B

c) C

d) D

e) B

f) D

g) A

4) A definição dos requisitos funcionais:

RF01	O sistema deve permitir o gerenciamento de funcionários e seus dados cadastrais
RF02	O sistema deve controlar o sistema de acesso de acordo com as permissões de cada ator.
RF03	Deve ser possível realizar o gerenciamento de horário dos funcionários.
RF04	O sistema deve possibilitar o lançamento de horários de consulta por meio de uma agenda médica.
RF05	Deve ser possível a consulta de horários marcados por médico, por data.
RF06	O sistema deve possibilitar o gerenciamento de Paciente (cadastro e ficha médica)
RF07	Deve ser possível incluir novos convênios ou mesmo excluir convênios com os quais a clínica opera.
RF08	É necessário que o sistema ofereça relatórios estatísticos de atendimento por convênio
RF09	É necessário que o sistema emita relatórios estatísticos de atendimento por área de especialização.

A definição dos atores:

Ator	Descrição	Responsabilidade
Médicos	Todos os 34 médicos possuem especialização, todos possuem conhecimentos básicos de informática e noção do funcionamento e procedimentos da clínica. A faixa etária se encontra entre 30 e 48 anos. Nenhum médico possui qualquer deficiência física.	Gerenciamento de Paciente
Atendente	As 2 atendentes possuem o segundo grau completo, razoável experiência em informática e conhecimento do processo de funcionamento da clínica. As atendentes estao na clínica a aproximadamente 3 anos.	Gerenciamento de horário dos funcionários Gerenciamento de agenda médica Permitir consulta de horários marcados por médico, por data. Gerenciamento de Paciente (cadastro)
Paciente	Pacientes com faixa etária até 14 anos, sob custódiaa e agendamento dos pais	Gerenciamento de agenda médica (marcar consulta)
Gerente	Médico especialista possui conhecimentos básicos de informática e noção do funcionamento e procedimentos da clínica. Possui 42 anos não apresenta nenhuma deficiência física.	Acesso a todas as funcionalidades.

Os diagramas de 3 casos de uso e a tabela de documentação do caso de uso:

Caso de uso: CSU001 Gerenciamento de Funcionário	
Breve descrição:	Gerenciamento do cadastro de dados pessoais e horários de trabalho de funcionários da clínica.
Ator primário:	Gerente
Ator secundário :	Funcionário
Pré-condições:	- O gerente estar devidamente logado no sistema.
Fluxo principal:	1. O gerente solicita uma inclusão de funcionário; 2. O gerente informa o nome do funcionário; 3. O sistema informa o código do funcionário; 4. O gerente informa os dados pessoais do funcionário; 5. O gerente informa os horários em que o mesmo estará na clínica(CSU; 6. O registro do funcionário é armazenado.
Fluxo alternativo e exceções:	No item 2, caso este nome já exista. Neste caso: 1.1 São apresentados os dados cadastrais do funcionário; 1.2 São apresentadas as possibilidades de Alterar, excluir ou finalizar.
Pós-condições:	Dados do funcionário armazenados
Requisito funcional:	RF01- O sistema deve permitir o gerenciamento de funcionários e seus dados cadastrais
Regras de negócio:	RN01- O funcionário do tipo médico só pode ser excluído se não houver nenhuma consulta agendada.

Caso de uso: CSU002 Agendamento de Horário Atendente CSU 002 Agendar horário <<extend>> CSU 005 Gerenciar Ficha Cadastral Paciente	
Breve descrição:	O caso de uso permite o agendamento do horario de consulta do paciente.
Ator primário:	Atendente
Ator secundário :	Paciente
Pré-condições:	- O atendente estar devidamente logado no sistema; - horários médicos disponibilizados.
Fluxo principal:	1. O paciente informa o nome do médico, data e horario desejado; 2. O atendente solicita a consulta para o médico informado nas datas solicitadas; 3. O atendente verifica se é primeira consulta na clínica a. Se sim executa o caso de uso Gerenciar Ficha Cadastral Paciente 4. O atendente solicita o tipo de consulta; 5. O agendamento do horário é armazenado. 6. A atendente informa novamente data, hora e médico da consulta para o Paciente.
Fluxo alternativo e exceções:	No item 1, caso o paciente não informe o nome do médico. Neste caso: A atendente sugere o nome de um dos médicos segundo sua especialidade. No item 2, caso não haja horário disponível para o médico. Neste caso: A atendente sugere outro horário, ou o nome de um segundo médicos segundo sua especialidade.
Pós-condições:	Consulta marcada.
Requisito funcional:	RF04- O sistema deve possibilitar o lançamento de horários de consulta por meio de uma agenda médica.
Regras de negócio:	RN02- se o tipo de consulta for retorno, agendar somente 10 minutos na agenda. Se for consulta normal agendar 20 minutos.

Caso de uso: CSU003 Cadastro de Convênios	
Gerente — CSU00 3 - Gerenciamento de Convenios	
Breve descrição:	Neste caso ocorre o gerenciamento dos convênios.
Ator primário:	Gerente
Ator secundário :	
Pré-condições:	O gerente deve estar devidamente logado no sistema;
Fluxo principal:	1- O gerente solicita uma inclusão de convênio; 2- É informado o nome do convênio 3- O sistema informa o código interno o convênio 4- São informados dados cadastrais do convênio 5- Os dados são armazenados.
Fluxo alternativo e exceções:	No item 2, caso este nome já exista. Neste caso: São apresentados os dados cadastrais do funcionário; São apresentadas as possibilidades de Alterar, excluir ou finalizar
Pós-condições:	Consulta marcada.
Requisito funcional:	RF07- Deve ser possível incluir novos convênios ou mesmo excluir convênios com os quais a clínica opera.
Regras de negócio:	

UNIDADE 6

1) As alternativas corretas são: **b** e **d**.

2) A seqüência é:

a) c

b) a

c) a

d) b

e) c

f) b

3) A seqüência é:

a) G

b) B

c) C

d) E

e) F

f) D

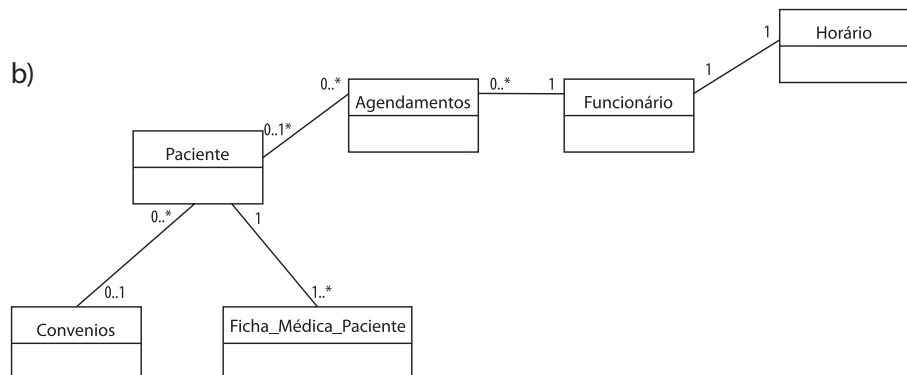
g) A

4)

a) As classes persistentes.

É possível identificar:

- A classe Paciente – que armazena os dados cadastrais do paciente
- A classe Agendamentos – que armazena o horário das consultas, nome do paciente e médico.
- A classe Funcionário – armazena os dados dos funcionários inclusive do funcionário Médico.
- A classe Horário – que armazena o horário de atendimentos da equipe médica
- A classe Ficha Médica – armazena a ficha de atendimento do paciente.

b)

UNIDADE 7

1) A seqüência correta é:

a) B

b) D

c) F

d) A

e) H

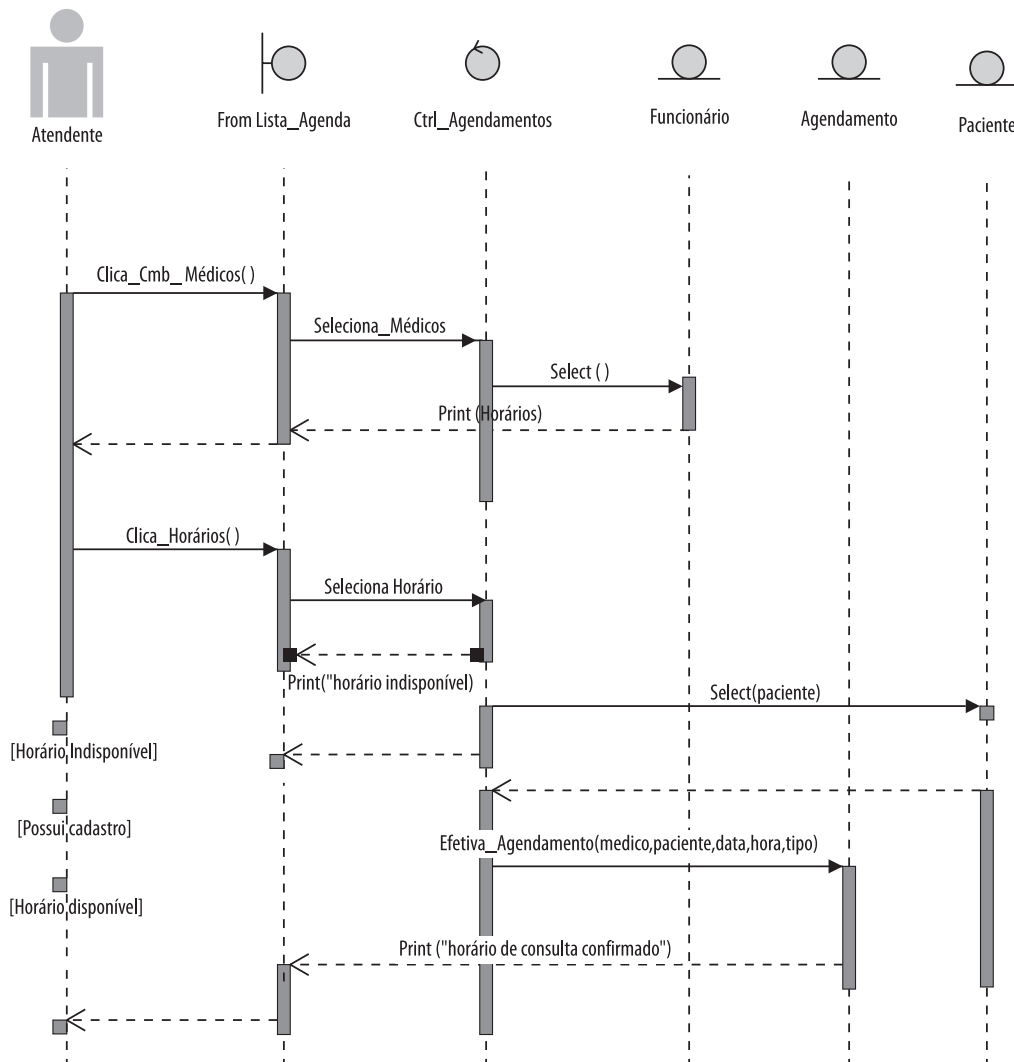
f) E

g) G

h) I

i) C

2) No diagrama a seguir, são descritos dois diagramas o Listar Agenda e o Agendamento do Horário.

CSU002- Agendamento de Horário

3) Alternativa correta: b.

UNIDADE 8

1) a) Esta modificação é chamada de transição entre estados.

b) Os estados representa o resultado de atividades executadas por um objeto.

c) O estado final indica o final do ciclo de vida de um objeto.

d) Os objetos de um sistema modificam seu estado de forma análoga a objetos do mundo real.

2) a) DTE

b) DA

c) DTE

d) DA

3) a) E

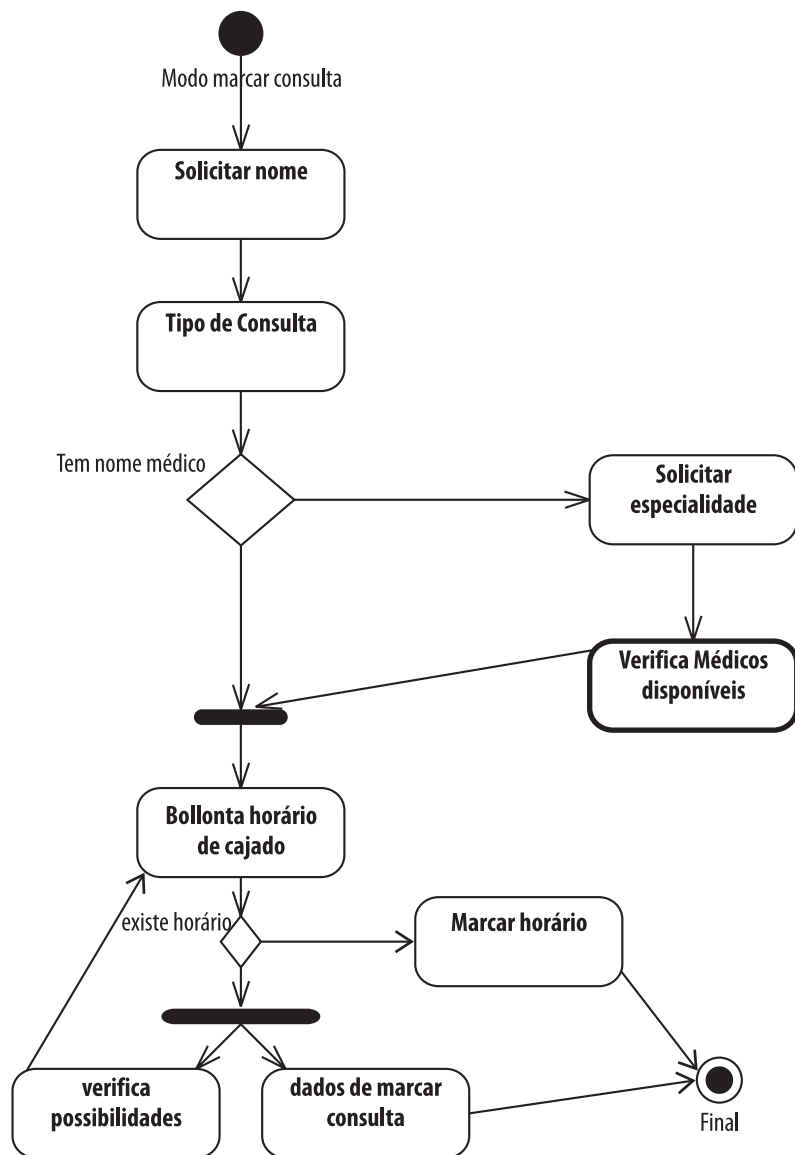
b) B

c) A

d) C

e) E

4) Diagrama de atividades do caso de uso Marcar Consulta da clínica Bem Estar a partir da visão do ator Atendente.

Modo marcar consulta
Solicitar nome
Tipo de Consulta
Tem nome médico
Solicitar especialidade
Verifica Médicos disponíveis
Bollonta horário de cajado
existe horário
Marcar horário
verifica possibilidades
dados de marcar consulta
Final

UNIDADE 9

1) As afirmativas corretas são: **b** e **d**.

2) a) A

b) C

c) A

d) C

e) B

f) B

3) a) E

b) A

c) C

d) D

e) B

4) O modelo utilizado será dependente do tipo de produto a ser desenvolvido e da dinâmica da empresa desenvolvedora, mas podemos citar 3 modelos que podem ser considerados interessantes em qualquer tipo de projeto, são eles:

- Modelo de domínio que procura estabelecer o contexto do sistema
- Modelo de caso de uso que estabelece os requisitos funcionais do sistema
- Modelo do processo que estabelece os mecanismos de concorrência e de sincronização do sistema.